Prozeßorientierte Entwicklung und Implementierung eines Interface-Systems zur PPS-Leitsystemkopplung in der Inselfertigung am Beispiel der Turbinenfertigung der ABB-Kraftwerke AG, Mannheim

Diplomhausarbeit

im Fachgebiet Wirtschaftsinformatik
am Lehrstuhl für Wirtschaftsinformatik und Informationsmanagement

Themensteller: Prof. Dr. Jörg Becker
Betreuer: Dipl.-Wirt. Ing. Christoph v. Uthmann

vorgelegt von: Jörg Evermann
Weststraße 36
49545 Tecklenburg
(01 72) 5 18 37 67

Abgabetermin: 27. März 1997

Inhaltsverzeichnis

Abkürzungsverzeichnis

ABB	Asea Brown Boveri
ANSI	American National Standards Institute
CAD	Computer Aided Drafting
CAM	Computer Aided Manufacturing
CAP	Computer Aided Planning
CAQ	Computer Aided Quality Assurance
CASE	Computer Aided Software Engineering
CIM	Computer Integrated Manufacturing
CNC	Computer Numerical Control
DBMS	Datenbank Management System
DDL	Data Definition Language
DIN	Deutsche Industrienorm
DML	Data Manipulation Language
DNC	Direct Numerical Control
DV	(Elektronische) Datenverarbeitung
EDV	Elektronische Datenverarbeitung
ERM	Entity Relationship Modell
FFS	Flexibles Fertigungssystem
FFZ	Flexible Fertigungszelle
FHM	Fertigungshilfsmittel
FLS	Fertigungsleitstand
FUD	Fertigungsunterstützende Daten
IS	Interface-System (Schnittstellen-System)
KW/P	ABB Kraftwerke AG, Bereich Produktion
KW/PG	ABB Kraftwerke AG, Bereich Generatorproduktion
KW/PM	ABB Kraftwerke AG, Bereich Produktion mechanischer Anlagen
KW/PT	ABB Kraftwerke AG, Bereich Produktion thermischer Anlagen
KW/PY	ABB Kraftwerke AG, Bereich Vertrieb/Ordermanagement
KWE	ABB Kraftwerke AG
MRP	Material Requirements Planning
NC	Numerical Control
PC	Personal Computer
PPS	Produktionsplanung und -steuerung
PRP	Project Requirements Planning
SPS	Speicherprogrammierbare Steuerung
SQL	Structured Query Language

Abbildungsverzeichnis

Tabellenverzeichnis

Motivation und Zielsetzung

Der erhöhte Wettbewerbsdruck zwingt viele Unternehmen, neue Techniken und Organisationsformen in der Fertigung einzusetzen. Kundenforderungen nach immer größerer Flexibilität lassen sich vielfach nicht mehr mit klassischen Fertigungsformen befriedigen. Vor diesem Hintergrund sieht sich auch die ABB Kraftwerke AG gezwungen, ihren Fertigungsbereich neu zu gestalten. Die klassische Werkstattfertigung wird aufgegeben und durch eine Fertigungsinselorganisation ersetzt, die mit Elementen der flexiblen Automatisierung realisiert ist.

Diese neue Organisationsform und die neuen Bearbeitungstechnologien stellen neue Anforderungen an Methoden und Instrumente für die Produktionsplanung und -steuerung. Diese Anforderungen lassen sich durch Leitstandskonzepte als eine konkrete Ausprägung dezentraler PPS-Systeme befriedigen, wie sie bei der ABB Kraftwerke AG zur Zeit eingeführt werden. Übergeordnet sorgt ein PPS-System für die Bereitstellung von Fertigungsauftragspools und Arbeitsplänen für die einzelnen Fertigungsinseln, die jeweils über einen eigenen Fertigungsleitstand verfügen, mit dessen Hilfe eine autonome Feinplanung und Steuerung durchgeführt werden soll.

Diese neuen Systeme müssen sowohl organisatorisch als auch technisch in ein sinnvolles Konzept integriert werden, um optimalen Nutzen zu erzielen, denn „nur eine integrierte Informationsbasis schafft die Möglichkeit, den Zeitraum zwischen der Äußerung des Kundenbedürfnisses und dessen Befriedigung zu verkürzen.“[1]. Diese Integration wird im Rahmen von CIM (computer integrated manufacturing) -Konzepten behandelt.

Ziel der vorliegenden Arbeit ist die Entwicklung und Implementierung eines Schittstellensystems. Dieses Schnittstellensystem soll die eingeführten Fertigungsleitstände mit dem PPS-System und dem NC-Programmiersystem informationstechnisch verbinden.

In den Kapiteln bis wird die vorliegende Arbeit in den fachlichen Kontext eingeordnet. Nach einer Einführung in die Konzepte der dezentralen Produktionsplanung und -steuerung werden Möglichkeiten der Verbindung, insb. durch Schnittstellensysteme, der entstehenden dezentralen Komponenten aufgezeigt. Daran schließt sich in Kapitel die Vorstellung der verwendeten Vorgehensweise bei der Entwicklung eines Schnittstellensystems an. Es werden Methoden vorgestellt, um dezentrale PPS-Komponenten in einen umfassenden Planungablauf einzubetten und informationstechnisch zu verbinden.

[1] Adam (FFS)(1993), S. 8.

Am Beispiel der Turbinenfertigung der ABB Kraftwerke AG wird eine solches Schnittstellensystem prototypenhaft entwickelt und implementiert werden. Durch die genaue Analyse der bestehenden Systeme in Kapitel werden die Rahmenbedingungen und Schwachstellen deutlich. Innerhalb dieser Rahmenbedingungen werden in Kapitel die fachlichen Anforderungen an das Schnittstellensystem entwickelt. Diese werden in Kapitel durch Prozeß- und Datendefinitionen des Schnittstellensystems formalisiert. Als mächtige Darstellungsmittel eignen sich dazu unter anderem ERM-Diagramme und Petri-Netze. Ausgehend von der Analyse der Anforderungen und der Definition der Schnittstellen wird ein erster evolutionärer Prototyp realisiert, der zur Demonstration der Funktionalität dient und weiteren Handlungsbedarf aufdecken soll.

Um eine spätere Pflege und Wartung sowie eventuelle Erweiterungen zu erleichtern wird soweit wie möglich auf CASE-Methoden und CASE-Werkzeuge zurückgegriffen. In der Realisierungsphase wird ein relationales Datenbankmanagementsystem (DBMS) für die Datenhaltung und die Programmiersprache Visual Basic für die Modulimplementierung verwendet.

Organisatorische Einordnung und fachliche Grundlagen

Die ABB Kraftwerke AG

Die ABB Kraftwerke AG (internes Firmenkürzel KWE) mit Sitz in Mannheim gehört mit ca. 3000 Mitarbeitern zur Asea Brown Boveri AG, Mannheim. Diese bildet den deutschen Teil des weltweit tätigen ABB Asea Brown Boveri Konzerns mit Hauptsitz in Zürich. Die Asea Brown Boveri AG erwirtschaftet einen Jahresumsatz von mehr als 10 Milliarden DM, Teil des weltweiten Konzernumsatzes von mehr als 27 Milliarden US$.[2]

Die ABB Kraftwerke AG tritt weltweit als Generalunternehmer, Konsortialführer oder -mitglied und als Produzent für komplette Dampf-, Gas- und Wasserkraftwerke und deren Komponenten auf. Zum Leistungsspektrum zählen neben den ingenieurtechnischen Planungsleistungen auch die Herstellung der Komponenten selbst und deren Lieferung und Montage.[3]

Aufbauorganisatorische Beschreibung der ABB Kraftwerke AG

Organisatorisch ist die KWE innerhalb des Gesamtkonzerns dem Geschäftsbereich Stromerzeugung zugeordnet. Auch in der deutschen ABB Gruppe von circa 60 selbständigen Gesellschaften, gehört sie, neben 10 weiteren Gesellschaften, dem Geschäftsbereich Stromerzeugung an.

Die KWE ist wiederum in 7 Geschäftsbereiche unterteilt, von denen der Bereich KW/P für die Produktion zuständig ist. KW/P fertigt in den Teilbereichen KW/PT und KW/PG die Kraftwerkskomponenten Dampf- und Gasturbinen bzw. Generatoren, sowohl für die ABB Kraftwerke AG als auch für andere Gesellschaften im ABB Konzern. Ein Teilbereich des KW/P, KW/PM, übernimmt die Herstellung der dafür benötigten mechanischen Baugruppen, insb. Gehäuseteile, Lager u.a.

Der Bereich KW/PM unterteilt sich wiederum in zwei Costcenter, Großmechanik und Klein- und Mittelmechanik, die neben den Fertigungsinseln auch ein eigenes Werkzeug-einstellcenter besitzen. Ihnen sind jeweils eigene NC-Programmierer und Arbeitsplaner zugeordnet.

Die Aufbauorganisationsstruktur der Asea Brown Boveri AG ist als Organigramm in Anhang in den relevanten Ausschnitten dargestellt.

---

[2] Vgl. o.V. ABB Jahresbericht, ABB (1993), S. 58ff.

[3] Vgl. o.V. Innovativ für Kraft und Wärme, ABB (1995), S. 2f.

Betriebstypologische Einordnung der ABB Kraftwerke AG

Die ABB Kraftwerke AG und insbesondere der Bereich KW/P läßt sich als typischer Auftragsfertiger im Investitionsgüterbereich beschreiben. Er fertigt komplette Generatoren und Turbinen für Kraftwerke mit allen ihren Komponenten. Der Fremdbezug beschränkt sich auf Rohteile, d.h. insb. bei der Turbinenproduktion auf Gußteile, die anschließend zu sehr komplexen Erzeugnissen weiterverarbeitet werden. Für Investitionsgüter in der betrachteten Größenordnung ist eine Serienfertigung oder Kleinserienfertigung nicht möglich. Aufgrund von kundenindividuellen Anforderungen besteht eine große Variabilität in den Produktspezifikationen. Die KWE hat sich aber erfolgreich bemüht, ihr Erzeugnisspektrum auf bestimmte Standarderzeugnisse zu begrenzen, die jeweils kundenspezifisch geändert werden. Aufgrund dieser Kundenspezifität der Erzeugnisse kann die Produktion und Disposition nur kundenauftragsorientiert durchgeführt werden. Es ist jedoch eine kleine Menge geringwertiger Erzeugnisse vorhanden, die in allen Varianten eingesetzt wird. Diese werden ohne Kundenbezug in kleinen Serien gefertigt. Die Betriebstypologie der ABB Kraftwerke AG läßt sich daher wie folgt einordnen:

Merkmal	Merkmalausprägungen			
Erzeugnis-spektrum	Erzeugnisse nach Kundenspezifikation	Typisierte Erzeugnisse mit kundenspezifischen Varianten	Standarderzeugnisse mit Varianten	Standarderzeugnisse ohne Varianten
Erzeugnis-struktur	Einteilige Erzeugnisse	Mehrteilige Erzeugnisse mit einfacher Struktur	Mehrteilige Erzeugnisse mit komplexer Struktur	
Auftragsauslösungsart	Produktion auf Bestellung mit Einzelaufträgen	Produktion auf Bestellung mit Rahmenaufträgen	Produktion auf Lager	
Dispositionsart	Disposition kundenauftragsorientiert	Disposition überwiegend kundenauftragsorientiert	Disposition überwiegend programmgesteuert	Disposition programmorientiert
Beschaffungs-art	Fremdbezug unbedeutend	Fremdbezug in kleinerem Umfang	Fremdbezug in größerem Umfang	Weitestgehender Fremdbezug

Fertigungsart	Einmalfertigung	Einzel- und Kleinserienfertigung	Serienfertigung	Massenfertigung
Fertigungs-ablaufart	Baustellenfertigung	Werkstattfertigung	Gruppen-/ Linienfertigung	Fließfertigung
Fertigungs-struktur	Fertigung mit geringer Tiefe	Fertigung mit mittlerer Tiefe	Fertigung mit großer Tiefe	

Quelle: Angelehnt an Kurbel(1993), S. 34.

Tab. 1.1: Betriebstypologie der ABB Kraftwerke AG

Das Projekt FFMA

Aufgrund der Kundenspezifität der Erzeugnisse bei Fertigung auf Kundenauftrag ist nur eine Einzel- oder Kleinserienfertigung möglich. Die kleinen Losgrößen machen eine häufige Umstellung der Maschinen auf ein anderes Produkt notwendig. Daraus entstehen sehr hohe Flexibilitätsanforderungen, insbesondere bezüglich der Produkt- und Mengenflexibilität. Um aber im Wettbewerb bestehen zu können, ist eine teilweise Automatisierung aus Kostengesichtspunkten unerläßlich. Mit einem Investitionsvolumen von mehr als DM 100 Mio. hat die ABB Kraftwerke AG im Rahmen des Projekts „*Flexible Fertigung Mannheim (FFMA)*“ im Bereich KW/PM Elemente der flexiblen Automatisierung eingeführt. Die Universalmaschinen in den Werkstätten wurden durch flexible Fertigungszellen (FFZ) und flexible Fertigungssysteme (FFS) abgelöst.

Zusammen mit den FFZ und FFS sind zwei Werkzeugeinstellcenter geschaffen worden, die mit computergestützten Werkzeugvoreinstellgeräten und EDV-gestützten Werkzeugverwaltungssystemen und Werkzeuglagern ausgestattet sind.

Die mechanische Fertigung des Produktionsbereiches KW/P, der Bereich KW/PM, ist im Rahmen des Projekts FFMA in sechs Fertigungsinseln umstrukturiert worden, die eine zentrale Ressource, das Sägezentrum, gemeinsam nutzen. Im folgenden werden nur die wesentlichen Betriebsmittel der Fertigungsinseln angegeben. Daneben existieren weitere Betriebsmittel wie z.B. einfache CNC-Maschinen und Bearbeitungsplätze für manuelle Tätigkeiten.

Fertigungsinsel „Großmechanik“

Die Fertigungsinsel „Großmechanik“ stellt Gehäuseteile für Gas- und Dampfturbinen durch spanende Bearbeitung aus angelieferten Gußteilen her. Die Werkstücke werden manuell mittels eines Portalkranes auf Lagerplätze oder direkt auf die Maschinen transportiert.

- Typische Losgröße 1
- Typische Bearbeitungszeit 5 - 320 h

Betriebsmittel:

- 2 Doppel-Gantry-Maschinen mit jeweils
    - 2 Bearbeitungsplätzen Plattenfeld
    - 2 Karussell-Bearbeitungsplätzen (Planscheiben)
    - 2 Kettenmagazinen mit jeweils 120 Plätzen für Werkzeuge

Auf den Plattenfeld-Bearbeitungsplätzen einer Doppel-Gantry können beide Portale arbeiten. Auf Karussell-Plätzen können nur die jeweils zugeordneten Portale arbeiten. Die Ausnahme bildet das Portal 2 der Doppel-Gantry-Maschine 1 welches zusätzlich auf dem Karussell-Platz von Portal 1 arbeiten kann.

- 1 Doppeltischmaschine
    - 1 Bearbeitungsplatz Plattenfeld
    - 1 Plattenfeld mit integriertem Karussell-Bearbeitungsplatz (Planscheibe)
    - 1 Kettenmagazin mit 120 Plätzen für Werkzeuge

Die beiden Doppel-Gantry Anlagen und die Doppeltischmaschine sind Bearbeitungszentren die sowohl Bohr- und Fräs- wie auch die Drehbearbeitung zulassen. Bei der Drehbearbeitung handelt es sich um das Senkrechtdrehen. Es kann daher nur auf den Karussellplätzen stattfinden. Für die Montage der Werkstücke stehen 2 Montageplätze zur Verfügung. Die Werkstücke können aber auch direkt auf den Plattenfeldern der Maschinen montiert werden.

Fertigungsinsel „Ergänzende Maschinen“

Diese Fertigungsinsel stellt als zweite Insel im Costcenter Großmechanik unterstützende Maschinen zur Verfügung, stellt also keine Fertigungsinsel im eigentlichen Sinn dar. Sie ist in der betrieblichen Praxis daher auch der Fertigungsinsel „Großmechanik“ untergeordnet.

Fertigungsinsel „Rotationsteil“

Die Fertigungsinsel „Rotationsteile“ stellt Teile für Futter und Wellen her:

- Typische Losgröße 1 bis 100
- Typische Teiledurchlaufzeit ca. 1 - 24 h

Der Transport der Werkstücke, der Werkzeuge und Vorrichtungen zwischen den Bearbeitungsplätzen, Magazinen und Lagerplätzen wird manuell durchgeführt.

Betriebsmittel:

- 1 Drehmaschine Heyligenstaedt 35
    - 1 Werkzeugstafette mit 7 Werkzeugen im Übergabebereich
- 1 Drehmaschine Georg Fischer
    - 1 Magazin mit 42 Plätzen für Werkzeuge

Fertigungsinsel „Kubische Teile“

Diese Fertigungsinsel stellt kubische Teile her:

- Typische Losgröße 1 bis 200
- Typische Teiledurchlaufzeit ca. ½ - 24 h

Betriebsmittel:

- 1 Bearbeitungszentrum MC60 (Burkhard + Weber)
    - 8 Pufferplätze für Paletten
    - 1 Werkzeugmagazin für 103 Werkzeuge

- 1 Bearbeitungszentrum MC 120 (Burkhard + Weber)
    - 1 Werkzeugmagazin für 168 Werkzeuge

Fertigungsinsel „Lager“

Diese Fertigungsinsel bearbeitet Werkstücke für Lager:

- Typische Losgröße 1 bis 5
- Typische Teiledurchlaufzeit ca. ½ - 24 h

Betriebsmittel:

- 2 SOLON Bearbeitungszentren (Dörries Scharmann) für Zerspanung mit jeweils
    - 2 Werkstück-Pufferplätzen
    - 2 Spindeln (eine davon feststehend)
    - 1 Werkzeugwechsler mit 2 Plätzen
    - verschiedenen Werkzeugplätzen in den Maschinen für insg. ca. 150 Werkzeuge

Die beiden Bearbeitungszentren sind mit einem schienengebundenen Transportfahrzeug verbunden. Werkstücke, Werkzeuge und Vorrichtungen werden auf Paletten gelagert und transportiert. Für den Transport innerhalb des flexiblen Fertigungssystems (FFS) werden transportsystemspezifische Paletten eingesetzt, für den Transport außerhalb werden Europaletten mit Untersatz verwendet.

Fertigungsinsel“ Schrauben und Wellen“

Diese Fertigungsinsel stellt Schrauben und Wellen aus gesägten Rohlingen her:

- Typische Losgröße 1 bis 500
- Typische Teiledurchlaufzeit ca. 1 min. - 2 h

Betriebsmittel:

- 2 Heynumat 15 Drehbearbeitungszentren mit jeweils

    - 2 Trommel-Werkzeugspeichern mit 12 Plätzen für Werkzeuge

- 1 Stangenautomat Heynumat 5

    - 1 Magazin mit 12 Plätzen für Werkzeuge

- 1 Zentrierstation, 1 Waschstation, 2 Meßschubladen, 2 Prüfstationen, 2 Prägestationen, 4 Wendestationen

Die Zentrierstation, die Prüf-, Präge-, Wende- und Waschstationen sowie die Meßschubladen werden durch eine zentrale SPS gesteuert.

- 1 Linearportal EPL-S 70 H NC (Promot)

    - 2 Laufwagen mit je 1 Wendegreifer für 1 Werkstück (max. 70 kg)

Das Linearportal mit den 2 Laufwagen verkettet beide Heynumat 15 Maschinen, die Zentrierstation, die Prüf-, Präge- und Wendestationen, die Waschstationen und die Meßschubladen. Werkstücke, Werkzeuge und Vorrichtungen werden auf Paletten gelagert und transportiert. Für den Transport innerhalb des FFS werden systemspezifische Paletten eingesetzt, für den Transport außerhalb werden Europaletten mit Untersatz verwendet.

Sägezentrum

Das Sägezentrum besteht aus einem Hochregallager für Stangenmaterial und einer Säge. Es stellt hauptsächlich das Ausgangsmaterial für die Fertigungsinsel „Schrauben und Wellen“ her. Die anderen Fertigungsinseln benötigen auch solches Material, allerdings in deutlich geringerem Umfang. Zum Sägezentrum gehört weiterhin eine Palettierstation, auf der die gesägten Stücke für die Fertigungsinsel „Schrauben und Wellen“ palettiert werden. Für jede Palette wird beim Palettieren der Rohlinge ein Palettenbelegungsplan erstellt und geführt. Die anderen Fertigungsinseln beziehen das Material unpalettiert.

Flexible Automatisierung und flexible Fertigungsinseln

Aufgrund von externen Faktoren, insb. dem Wandel der Absatzmärkte von ungesättigten Verkäufer- zu gesättigten Käufermärkten, werden Forderungen nach einem variantenreichem und qualitativ hochwertigem Erzeugnisspektrum mit hoher Termintreue bei kurzen Lieferzeiten lauter. Daraus läßt sich die Forderung nach einer

flexiblen Fertigung ableiten, die andererseits aber auch produktiv und kostengünstig sein muß.

Flexibilität in der Fertigung läßt sich nach Förster (1988) durch fünf Flexibilitätsarten beschreiben:

Flexibilitätsart	Qualitäts-Merkmale
Produktflexibilität	Die Fähigkeit eines Systems, ein vorgegebenes Werkstückspektrum in beliebiger Reihenfolge zu bearbeiten.
Änderungsflexibilität	Die Anpassungsfähigkeit eines Systems an neue oder geänderte Teile und Fertigungsverfahren.
Mengenflexibilität	Die Fähigkeit eines Systems, bei unterschiedlichen Auslastungsgraden wirtschaftlich zu arbeiten.
Fertigungsredundanz	Das Vorhandensein mehrerer sich entsprechender, alternativ zueinander einsetzbarer Funktionselemente.
Integrationsflexibilität	Die Möglichkeit, ein System durch Ergänzung einzelner Funktionselemente zu erweitern oder mit anderen Systemen zu verketten.

Quelle: Angelehnt an Förster (1988), S. 6.

Tab. 1.2: Arten der Flexibilität

Flexible Automatisierung

Die klassische Werkstattfertigung zeichnet sich durch das Prinzip hochgradig arbeitsteiliger Verrichtungsorientierung aus und stellt das Betriebsmittel als Planungsobjekt in den Vordergrund.[4] Durch den Einsatz von Universalmaschinen verfügt sie zwar über sehr große Flexibilität in allen fünf genannten Arten, diese wird aber durch einen Mangel an Automatisierungspotential erkauft. Daher kann sie den gestellten Forderungen nicht gleichzeitig gerecht werden.[5]

Aufgrund des relativ hohen Rüstzeitanteils an den Durchlaufzeiten in Werkstätten mit Universalmaschinen, wird versucht, diese durch verschiedene Formen der *flexiblen Automatisierung* zu ersetzen. Die dabei eingesetzten Maschinen versuchen durch das Zusammenfassen von klassisch getrennten Arbeitsvorgängen die Anzahl und die Dauer

[4] Vgl. Scheer (1994), S. 297.
[5] Vgl. Becker, Rosemann (FSS) (1993), S. 56.

der Rüstvorgänge zu verringern. Sie zielen auf einen Kompromiß im Zielkonflikt zwischen Flexibilität und Automatisierung.[6] Nicht mehr die reine Bearbeitungsleistung ist entscheidend, vielmehr soll die Gesamtbearbeitung von Teilen möglichst ohne unproduktive Umrüstvorgänge durchführbar sein. Dabei soll eine schnelle Umstellung auf andere Produkte oder Varianten im Auftragsmix möglich sein. Ausgehend von diesen Anforderungen haben sich verschiedene neue Formen der Bearbeitungsmaschinen und
-gruppierungen entwickelt.[7] Basis aller dieser Ausprägungen der *flexiblen Automatisierung* ist die computergesteuere Werkzeugmaschine (CNC-Maschine).

Bearbeitungszentren

Bearbeitungszentren sind computergesteuerte Maschinen, die Werkstücke innerhalb einer Aufspannung auf mehrere, klassisch getrennte, Arten bearbeiten können, z.B. kombinierte Bohr- und Fräszentren. Bearbeitungszentren verfügen daher über Werkzeugmagazine, so daß Werkzeuge während des Betriebes automatisch gewechselt werden können. Zumeist ist während der Bearbeitung ein Werkzeugwechsel im Werkzeugmagazin möglich.[8] Bearbeitungszentren verfügen damit über Systeme für die Funktionen der

- Werkstückbearbeitung,
- Werkzeugspeicherung,
- Werkzeughandhabung und der
- Werkzeugver- und -entsorgung.

Flexible Fertigungszellen

Eine flexible Fertigungszelle (FFZ) besteht aus einem Bearbeitungszentrum, ergänzt um automatischen Werkstücktransport.[9] Sie erweitert das Bearbeitungszentrum also um die Funktionen für die

- Werkstückspeicherung und die

---

6 Vgl. Förster (1988), S.5.
7 Vgl. Becker, Rosemann (CIM) (1993), S. 252ff.
8 Vgl. Scheer (1994), S. 353f.
9 Vgl. Arning (1987), S. 70.

- Werkstückhandhabung

und ermöglicht so zusätzlich zu den Funktionen des Bearbeitungszentrums das hauptzeitparallele Rüsten und, durch automatische Werkstückzufuhr, bedienerarmen oder bedienerlosen Betrieb über einen längeren Zeitraum.[10]

Flexible Fertigungssysteme

Ein flexibles Fertigungssystem (FFS) besteht aus einer Gruppe programmgesteuerter Maschinen, normalerweise Fertigungszellen, die über ein Transport- und ein Informationssystem mit einem führenden Rechner verbunden sind. Dieser Rechner übernimmt die Steuerung der gesamten Anlage ebenso wie die Bereitstellung der NC-Programme für die Maschinen. Es liegt damit ein DNC-Betrieb vor (Direct Numerical Control)[11]. Die Werkstücke können ein FFS in unterschiedlichen Pfaden durchlaufen. Ebenso wie mit einer FFZ ist mit einem FSS ein bedienerarmer oder bedienerloser Betrieb über einen längeren Zeitraum möglich. Als Transportsysteme innerhalb eines FFS sind häufig schienengebundene Transporteinrichtungen, insb. für Paletten, Roboter oder induktiv geführte Transportfahrzeuge im Einsatz.[12]

Unter Fertigungshilfsmitteln (FHM) werden Werkzeuge, Vorrichtungen, Meß- und Prüfmittel, NC-Programme u.a. verstanden.[13] Eine Klassifizierung in physische und logische Fertigungshilfsmittel liegt nahe. Im folgenden sei unter Fertigungshilfsmitteln nur die Menge der physischen Mittel verstanden, z.B. Werkzeuge und Vorrichtungen, aber auch Skizzen und Zeichnungen, der Rest soll in dieser Arbeit unter dem Begiff fertigungsunterstützende Daten (FUD) bekannt sein. Dieses sind insb. NC-Programme, Zeichnungen, Skizzen, Aufspannbeschreibungen, etc. in elektronischer Form.

Um die gewünschte Flexibilität im Einsatz der FFZ und FFS realisieren zu können, sind in den Bearbeitungzentren ständig wechselnde Kombinationen von Fertigungshilfsmitteln, insb. Werkzeugen, notwendig. Daher sind Funktionen für die Werkzeugbevorratung und die Werkzeugbereitstellung erforderlich[14]. Die Werkzeugbereitstellung umfaßt den Zusammenbau von Werkzeugen aus Komponenten, die Vermessung der Komplettwerkzeuge sowie deren Transport und Einlagerung in die Maschinen.

---

10 Vgl. Förster (1988), S. 7.
11 Vgl. Scheer (1994), S. 354
12 Vgl. Förster (1988), S. 73.
13 Vgl. Förster (1988), S. 10.
14 Vgl. Förster (1988), S. 92.

Gruppenfertigung

Allen drei genannten Ausprägungen der flexiblen Automatisierung gemein ist die Reintegration der Arbeit durch das Zusammenfassen von Bearbeitungsschritten. Alle stellen das Werkstück, nicht wie bei den klassischen Maschinen die Verrichtung, in den Vordergrund der Betrachtung. Die Orientierung am Objekt der Bearbeitung, verbunden mit dem Ziel der Komplettbearbeitung in FFZ und FFS, steht in engem Zusammenhang mit gruppentechnologischen Prinzipien[15]. Diese umfassen die

- „Zusammenfassung fertigungsähnlicher Teile zu Teilefamilien,
- objektorientierte Betriebsmittelanordnung,
- Arbeitserweiterung durch Bildung einer Arbeitsgruppe,
- Aufgabendelegation in die Arbeitsgruppe.“ [16]

Die flexible Automatisierung steht damit in einem engen Zusammenhang zu organisatorischen Konzepten, die ähnlichen Prinzipien folgen, insbesondere den Fertigungsinseln.

Flexible Fertigungsinseln

Flexible Fertigungsinseln, im folgenden einfach Fertigungsinseln (FI) genannt, sind teilautonome, am Objektprinzip ausgerichtete Organisationseinheiten in denen das Erzeugnis bzw. die Baugruppe als Planungsobjekt im Vordergrund steht. Fertigungsinseln sind in der Weise abgeschlossene Bereiche, daß in einer Fertigungsinsel eine Anzahl gleicher oder ähnlicher Produkte möglichst vollständig gefertigt werden kann. Die zur Bearbeitung notwendigen Maschinen sind dabei räumlich zusammengefaßt. [17] Die Mitarbeiter der Fertigungsinsel tragen als Arbeitsgruppe oder Team die Ergebnisverantwortung für die von ihnen erzeugten Produkte. Diese Verantwortung bezieht sich sowohl auf die Qualität der Erzeugnisse als auch auf deren termingerechte und möglichst kostenminimale Herstellung. [18] Kennzeichnend für die Fertigungsinsel ist weiterhin die Integration vieler Tätigkeiten der Fertigungsvorbereitung in die Insel und die, auf technischer Seite durch flexible Automatisierung realisierte, Rückführung der Arbeitsteilung.[19] Obwohl die Betriebsmittel einer Fertigungsinsel in der Praxis häufig als flexible Fertigungssysteme

15 Vgl. Becker, Rosemann (FFS) (1993), S. 57.
16 Becker (FFS) (1993), S. 58.
17 Vgl. Keller, Kern (1991), S. 110f.
18 Vgl. Becker (1993), S. 46.
19 Vgl. Arning (1987), S. 78f.

oder flexible Fertigungszellen ausgelegt sind, ist dieses aber nicht zwingend erforderlich. Es ist durchaus eine überwiegend manuelle Bearbeitung in Fertigungsinseln vorstellbar.[20] Notwendig ist nur, daß die bearbeiteten Werkstücke gleiche Bearbeitungsmaschinen benötigen, wobei durchaus unterschiedliche Bearbeitungsreihenfolgen erlaubt sind.

Die Inselfertigung stellt damit eine Ausprägung der Gruppenfertigung mit den in aufgeführten Prinzipien dar. Fertigungsinseln sollen Rationalisierungspotentiale nicht mehr durch Optimierung der einzelnen Bearbeitungsvorgänge realisieren, sondern dieses durch einen *ganzheitlichen Blick* auf den Fertigungsprozeß einer Teilefamilie tun. Durch die physikalische Gruppierung der Betriebsmittel und den evtl. automatisierten Transport der Werkstücke und Werkzeuge soll der zeit- und kostenmäßige Aufwand für die Handhabung und Lagerung von Werkstücken sowie für Rüstvorgänge gegenüber der Werkstattfertigung drastisch reduziert werden.[21] Die Begrenzung auf einen kleinen Ausschnitt der Produktion, die Teilefamilie, fördert die Übersichtlichkeit vor Ort und ermöglicht so eine bessere Planung und Steuerung der Fertigung innerhalb der Insel. Durch den so möglichen Abbau der organisationsbedingten Liegezeiten von Werkstücken ergeben sich dadurch kürzere Durchlaufzeiten und bessere Termintreue für die Erzeugnisse. Andererseits werden durch kürzere Liegezeiten die Pufferbestände zwischen den Bearbeitungsmaschinen reduziert, wodurch sich eine geringere Kapitalbindung im Umlaufvermögen für das Unternehmen ergibt.

Fertigungsinseln nutzen einerseits die Synergieeffekte ähnlicher Produkte durch die Bildung von Teilefamilien aus („*economies of scale*“), zum anderen lassen sie aber durch die erreichte hohe Fertigungstiefe, im Idealfall die Komplettbearbeitung, einen geringen Koordinationsaufwand entstehen („*economies of scope*“).

Dezentrale Planung in Fertigungsinseln mit Leitstandsystemen

Von der Gewinnmaximierung als oberstes Unternehmensziel wird die Maximierung der Wirtschaftlichkeit, definiert als Quotient aus Leistung und Kosten, als Unterziel abgeleitet. Die PPS beschränkt sich auf die Betrachtung des Nenners[22]. Gegenstand der PPS ist damit die kostenminimale Gestaltung der betrieblichen Leistungserstellung, also der Bereitstellung von Gütern in bestimmten Mengen zu vorgegebenen Terminen in verlangter Qualität.[23] Sie versucht dies durch die Beantwortung der Frage wann, was, in

20 Vgl. Scheer (1994), S. 355.
21 Vgl. Adam (PRODUKTION) (1993), S. 11f.
22 Vgl. Kurbel (1993), S. 18f.
23 Vgl. Westkämper, Wiedenmann (1996).

welcher Menge auf welchem Betriebsmittel zu fertigen ist, so daß die entstehenden Kosten minimal werden.[24]

Ausgehend von diesem Ziel sind zahlreiche Modelle entwickelt worden, mit denen die gestellten Fragen beantwortet werden sollen. Als besonders problematisch haben sich dabei konfliktäre Unterziele und interdependente Planungsbereiche herausgestellt. Diese Planungsbereich dürfen nicht isoliert betrachtet werden, vielmehr muß versucht werden, gesamtheitlich optimale Pläne zu erzeugen. Ausgehend von dieser Erkenntnis sind zahlreiche Simultan- bzw. Totalmodelle entwickelt worden, die in der Praxis allerdings an mangelnder Rechenbarkeit aufgrund der enormen Anzahl zu berücksichtigender Variablen scheiterten.[25] Um die große Komplexität, verbunden mit dem hohen Datenvolumen, dennoch praktisch handhabbar zu machen, wurde versucht, die einzelnen Planungsbereiche der PPS, insbesondere die Mengen- und Zeitplanung, zu entkoppeln und getrennt zu behandeln. Die so entstehenden Einzelziele jedes Planungsbereiches laufen jedoch Gefahr, als Ganzes nicht kompatibel mit dem übergeordneten Ziel der PPS zu sein.[26] Eine Verbesserung zur vollständig getrennten Behandlung der Teilziele stellen hierarchische Modelle dar, die die Teilbereiche der Planung in eine Hierarchie einordnen und somit einen Teil der Interdependenzen berücksichtigen können. Eine übergeordnete Planungsebene macht Vorgaben für die untergeordneten Ebenen und kann von dort Rückkopplungen erhalten. Die Aufteilung der Planungsebenen erfolgt oft nach dem zeitlichen Planungshorizont, d.h. auf oberster Ebene befinden sich die Bereiche mit dem längsten Planungshorizont.[27]

Ein in der Praxis sehr häufig anzutreffendes hierarchisches Modell ist das MRP-Modell (Material Requirements Planning, Materialbedarfsplanung). Sukzessiv werden dabei die folgenden Funktionen durchlaufen:

- Primärbedarfsplanung,
- Sekundärbedarfsplanung, insb. Stücklistenauflösung,
- Losbildung und Vorlaufverschiebung,
- Durchlaufterminierung mit anschließendem Kapazitätsabgleich,
- Fertigungsauftragsfreigabe und die
- Maschinenbelegungsplanung.[28]

[24] Vgl. Kurbel (1993), S. 18f.
[25] Vgl. Kurbel (1993), S. 39-47ff.
[26] Vgl. Adam (PLANUNG), S. 405.
[27] Vgl. Kurble (1993), S. 46f.
[28] Vgl. Adam (PRODUKTION) (1993), S. 454.

Dabei schränken die Planungsergebnisse der vorgelagerten Funktionen den Entscheidungsraum der nachfolgenden Planungsfunktionen in der Weise ein, als daß sie als verbindliche Vorgaben gelten. Rückkopplungen zwischen den Hierarchiestufen sind im MRP-Modell nicht vorgesehen.[29]

Klassische PPS-Systeme orientieren sich stark am hierarchischen MRP-Modell der Produktionsplanung. Sie sind in den sechziger und siebzieger Jahren für die Unterstützung der mittelfristigen Disposition mit dem Schwerpunkt auf die Materialwirtschaft entwickelt worden. Sie richten sich stark an kaufmännischen Anforderungen aus und unterstützen die technischen Funktionen, wie z.B. die Produktentwicklung oder die Konstruktion, kaum. Ebensowenig wie die planerischen technischen Bereiche werden die fertigungsnahen und ausführenden technischen Bereiche, also z.B. Maschinensteuerung, Transportsteuerung, etc., unterstützt.

Nachteilig ist insbesondere diese fehlende Nähe zur eigentlichen Fertigung. Klassische Systeme planen auf allen Dispositionsebenen mit der gleichen Granularität. Es erfolgt z.B. auch im mittelfristigen und langfristigen Bereich der Sekundärbedarfsplanung eine minutengenaue Planung auf Arbeitsgangebene. So entstehen zum einen unrealistische Pläne, die von den Mitarbeitern nicht angenommen werden und an denen manuell „vorbeigeplant“ wird. Andererseits wird der benötigte Planungsaufwand so groß, daß eine ständig neue Planung nicht möglich ist. Zeitkritische Daten, z.B. über Betriebsmittelstörungen, Eilaufträge etc. gehen damit nicht, oder nur sehr spät, in eine geänderte Planung ein.[30]

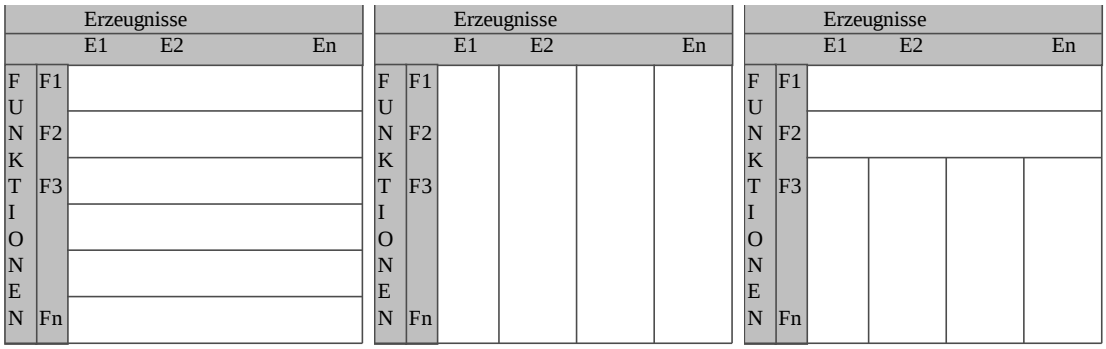
Aufgrund der Mängel sowohl der klassischen Systeme als auch der diesen zugrundeliegenden Modelle, wird in der PPS nach Konzepten gesucht, die diese Mängel beseitigen. Eines dieser Konzepte ist die Dezentralisierung der PPS.

Produktorientierte Dezentralisierung

Die Komplexität der PPS kann durch die Anzahl von Erzeugnissen und die Anzahl der dafür auszuführenden Funktionen dargestellt werden. Die Komplexitätsreduktion muß durch eine geeignete Aufteilung dieses Raumes geschehen. Geeignet heißt, daß zwischen den Teilen dieses Raumes möglichst geringe Abhängigkeiten bestehen, innerhalb der Teile aber eine starke Kopplung gegeben ist. Klassische PPS-Systeme zerlegen diesen Komplexitätsraum funktional (vgl. Abb. 1.1a) und orientieren sich damit an der bislang dominierenden Form der Werkstattfertigung. Dadurch lassen sich innerhalb einer Planungsfunktion die Interdependenzen zischen den Teilefamilien, z.B.

[29] Vgl. Adam (PRODUKTION) (1993), S. 456.
[30] Vgl. Kurbel (1993), S. 30f.

durch Losbildung, berücksichtigen. Außerachtgelassen werden Interdependenzen zwischen den Planungsfunktionen.[31]



Quelle: Angelehnt an Scheer (1991), S. 14.

Abb. 1.1: Möglichkeiten der Komplexitätsreduzierung

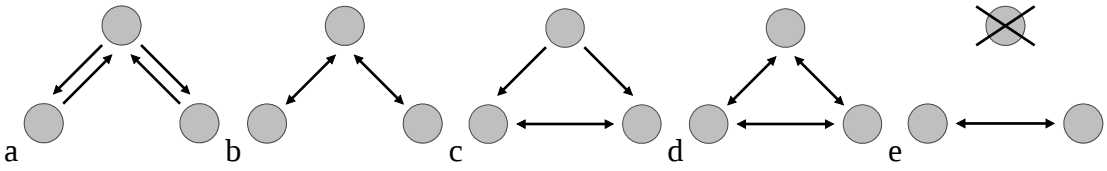
Diese Art der Zerlegung ist aber nicht die einzig mögliche, auch eine Zerlegung nach Erzeugnissen ist denkbar (vgl. Abb. 1.1b). Durch die Beschränkung auf ein kleines Teilespektrum und eine geringere Anzahl von Betriebsmitteln läßt sich die Planungskomplexität auf praktisch handhabbare Größenordnungen reduzieren. So lassen sich die Interdependenzen zwischen den Planungsfunktionen innerhalb einer Teilefamilie berücksichtigen. Durch die geringere Anzahl von Planungsvariablen und die Produktspezifität der Planungen wird eine größere Fertigungsnähe und so eine aktuellere und flexiblere Planung erreicht.[32]

[31] Vgl. Scheer (1991), S. 15.
[32] Vgl. Scheer (1991), S. 15.

Diese Zerlegung nur nach Teilefamilien zerstört Synergieeffekte zwischen den Erzeugnissen, die Zerlegung rein nach Funktionen berücksichtigt keine Interdependenzen zwischen diesen. Es sind aber auch Mischformen der Zerlegung denkbar (vgl. Abb. 1.1a). Dadurch lassen sich die Abhängigkeiten zwischen Planungsfunktionen und Teilefamilien sehr gut berücksichtigen. Bei dieser Form der Komplexitätsreduzierung entsteht ein sehr hoher Kommunikationsbedarf zwischen den übergeordneten, funktional gegliederten Bereichen und den dezenralen, produktorientiert gegliederten Bereichen. Zur Deckung dieses Bedarfes können verschiedene Formen der Kommunikation eingesetzt werden.[33]

Koordination dezentraler Planungsbereiche

Grundsätzlich sind verschiedene Mechanismen zum Informationsaustausch und zur Koordination von einer dezentralen Einheit mit anderen dezentralen Einheiten und dem übergeordneten Planungsbereich denkbar.



Quelle: Westkämper, Wiedenmann (1996), S. 40.

Abb. 1.2: Formen der Koordination

Detailplanung Top-Down

Bei dieser Form der Koordination macht die übergeordnete Planungsebene Vorgaben, die von den untergeordneten Ebenen einzuhalten sind (Abb. 1.2 a). Diese machen wiederum Rückmeldungen über den Grad der Planerfüllung. Vorstellbar ist aber, daß die übergeordnete Ebene die Gegebenheiten der untergeordneten Ebenen bei der Planung bereits antizipiert (Feed-Forward-Rückkopplung). Werden die Annahmen, die auf der oberen Ebene über die untergeordneten Ebenen gemacht werden, aufgrund der durch die Planausführung aufgetretenen Planabweichungen geändert, so spricht man vom Ex-post-feedback[34]. Diese Form der Dezentralisierung ist heute durch den Einsatz zentraler PPS
-Systeme und dezentraler Fertigungsleitstände in vielen Betrieben realisiert.

[33] Vgl. Scheer (1991), S. 15f.
[34] Vgl. Adam (PLANUNG), S. 322ff.

Vertikale Regelung

Es ist aber auch vorstellbar, daß die untergeordneten Planungsebenen Informationen an die übergeordnete Ebene zurückgeben, aufgrund derer die Planung geändert wird (Abb. 1.2 b). Dieses kann solange passieren, bis die Planvorgaben von den untergeordneten Ebenen eingehalten werden können, oder andere Bedingungen erfüllt sind (Ex-ante-feedback). Erst dann wird der Plan in den untergeordneten Ebenen ausgeführt.[35]

Grobplanung mit horizontaler Koordination

Bei dieser Form der Koordination macht die übergeordnete Ebene Planvorgaben und die untergeordneten Ebenen verständigen sich untereinander, um die Vorgaben zu erfüllen (Abb. 1.2 c). Eine solche horizontale Koordination entlastet die übergeordnete Planungsebene. Dort müssen nur noch planungsbereichsübergreifende Vorgaben erstellt werden, die dann an alle untergeordneten Ebenen gleichermaßen gegeben werden. Durch horizontale Koordination lassen sich auch *künstliche Engpässe* vermeiden, die insb. bei der Top-Down-Planung entstehen. Dort muß den Arbeitsgängen, mit denen die dezentralen Einheiten beauftragt werden, immer ein großzügiger Zeitpuffer mitgegeben werden. Dieser Puffer ist notwendig, um zu verhindern, daß ein Arbeitsgang, der in einer dezentralen Einheit verspätet ist, die Planung in nachfolgenden Einheiten zunichte macht. Da bei einer horizontalen Koordination Meldungen über Störungen sofort weitergegeben werden können, lasssen sich solche Verzüge planerisch ausgleichen, indem sich die dezentralen Einheiten untereinander verständigen.[36] Erste detailierte Konzepte für die Planung mit horizontaler Kommunikation existieren bereits, z.B. ein System von kooperierenden Leitständen[37]

Vertikale Regelung mit horizontaler Koordination

Auch mit horizontaler Koordination ist ein Regelkreis zwischen den Hierarchieebenen vorstellbar (Abb. 1.2 d). Sollten einzelne Vorgaben der übergeordneten Ebene nicht erfüllbar sein, so kann der Plan vor Ausführung auf der übergeordneten Ebene revidiert werden.

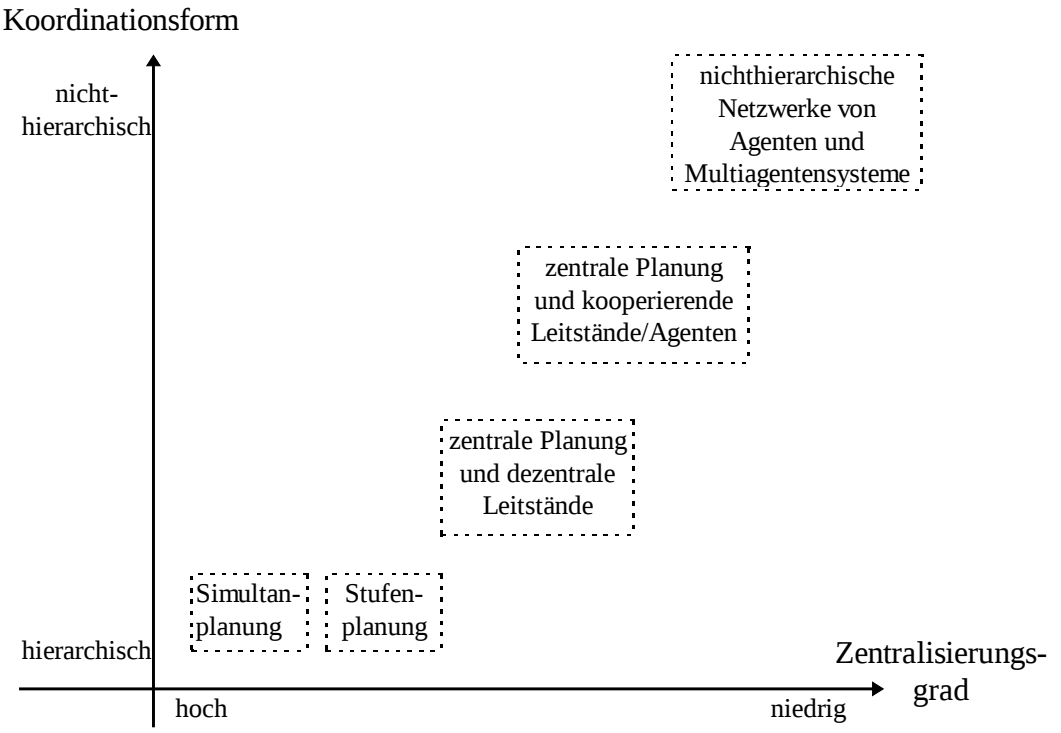
Rein horizontale Koordination

Eine rein horizontale Koordination ohne übergeordnetes Planungssystem ist möglich (Abb. 1.2 e) wenn mindestens ein System in der Lage ist, nicht nur mit anderen Planungssystemen, sondern auch mit der Außenwelt zu kommunizieren. Neuere

35 Vgl. Adam (PLANUNG), S. 322ff.
36 Vgl. Westkämper, Wiedenmann (1996).
37 Vgl. Schneider, Rinschede (1996).

Ansätze die dezentrale Planung mittels Agenten- und Multiagentensystemen oder Netzwerken von Agenten[38] zu unterstützen, fallen in diese Kategorie der rein horizontalen Planung.

Die genannten Beispiele dezentraler PPS-Konzepte lassen sich nach den Kriterien Zentralisierungsgrad und Hierarchisierungsgrad einordnen. Je geringer der Zentralisierungsgrad und die Anzahl der Hierarchiestufen ist, desto mehr eignet sich die horizontale Kommunikation für die Koordination der Systemkomponenten.



Quelle: Ferstl, Mannmeusel (1995)

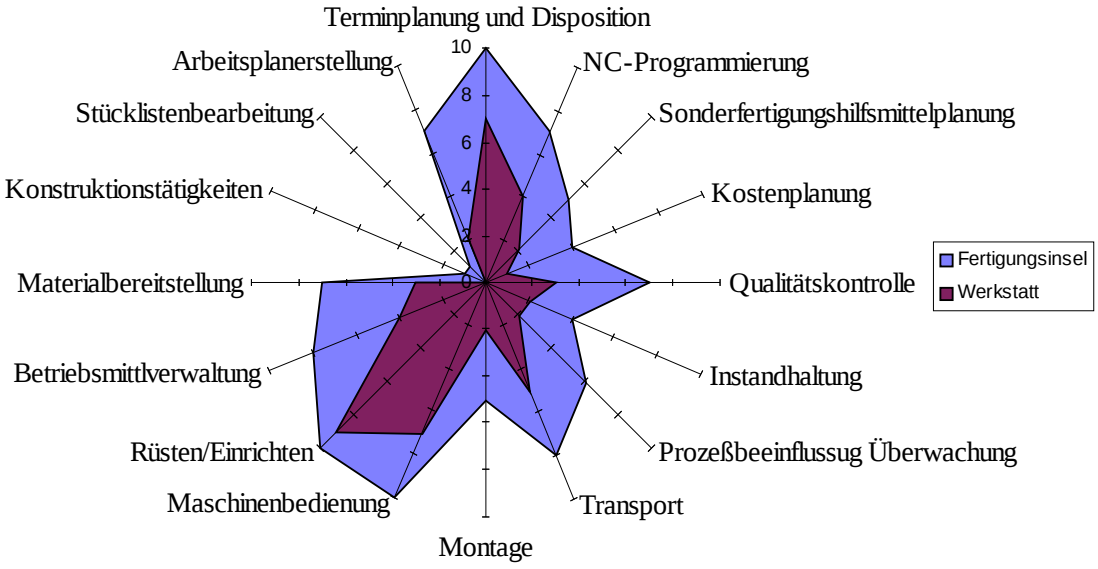
Abb. 1.3: Entwicklungstendenzen der PPS

Durch die Aufteilung der Fertigung in teilautonome Fertigungsinseln kann auch eine Aufteilung der zentralen Produktionsplanung und -steuerung (PPS) in teilautonome Planungsbereiche erfolgen, die jeweils einer Fertigungsinseln und damit einer Teilefamilie zugeordnet sind.

[38] Vgl. Ferstl (1995).

Die Fertigungsinsel als dezentraler Planungsbereich

Die Möglichkeit der weitgehenden Komplettbearbeitung einer Teilefamilie macht die Fertigungsinsel materialflußtechnisch sehr stark unabhängig vom Rest des Betriebes. Diese Unabhängigkeit führt zu einer Menge möglicher *Autonomiegrade* der Fertigungsinsel, die in der folgenden Abbildung kurz dargestellt und denen einer zentral gesteuerten Werkstatt gegenübergestellt sind:



Quelle: Maßberg (1993), S. 13.

Abb. 1.4: Autonomiegrade der Fertigungsinsel

Da jeder Materialfluß einen Informationsfluß in den Planungssystemen auslöst, kann die Fertigungsinsel auch informationstechnisch zu großen Teilen als unabhängig betrachtet werden. Dies hängt insbesondere auch vom Grad der Integration der vor- und nachgelagerten dispositiven Funktionen in die Insel ab. Die damit gegebene planerische Autonomie besteht sowohl in horizontaler als auch in vertikaler Richtung.

Die horizontale Unabhängigkeit bedeutet eine Beschränkung der Planung auf überschaubare Ausschnitte aus der gesamten Fertigung, nämlich die Fertigungsinsel. Je geringer die materialflußtechnischen Interdependenzen zwischen Fertigungsinseln sind, um so mehr kann in der Fertigungsinsel eine eigene Produktionsplanung gemacht werden, denn die Planung muß auf Beschränkungen durch inselexterne Betriebsteile keine Rücksicht nehmen. Damit ist die horizontale Planungsautonomie um so größer je geringer materialflußtechnische Beziehungen zu anderen Teilen der Produktion existieren. Die vertikale Autonomie der Fertigungsinsel bezieht sich auf die

Informationsmenge, die zwischen einem übergeordneten, zentralen Planungssystem und den jeweiligen Fertigungsinseln ausgetauscht wird. Im Idealfall beschränkt sich der Informationsfluß vom Planungssystem an die Insel auf Fertigungsaufträge mit Terminvorgaben. Diese Termine können z.B. auf bekannten oder geschätzten Durchlaufzeiten beruhen. Die Fertigungsinsel hat somit sehr viel Dispositionsspielraum für die Feinplanung auf der Ebene von einzelnen Betriebsmitteln bzw. auf der Ebene der einzelnen Arbeitsgänge. Sie meldet dementsprechend auch nur die Fertigstellung von Aufträgen an die übergeordnete Dispositionsebene zurück.[39]

Auf beiden Planungsebenen laufen damit ähnliche Vorgänge ab. Sie unterscheiden sich nur bezüglich der Granularität der betrachteten zeitlichen Intervalle, der Planungseinheit und der Kapazitätseinheiten.[40]

	Übergeordnete Planungsebene	Fertigungsinsel
kleinster planbarer Vorgang	Fertigungsaufträge	Auftragspositionen
kleinste planbare Kapazitätseinheit	Fertigungsinseln	Maschinen (evtl. auch Werkzeuge etc.)
zeitliches Raster	tages- oder wochengenau	minuten- oder stundengenau
zeitlicher Horizont	Wochen/Monate	Tage/Wochen

Quelle: Angelehnt an Förster (1988), S. 91.

Tab. 1.1: Planungsobjekte in hierarchischen Planungsebenen

Obwohl die Fertigungsinsel einen hohen Grad an Unabhängigkeit besitzt, darf der Planungsbereich Fertigungsinsel nicht losgelöst vom Rest der Produktion gesehen werden, vielmehr müssen die Pläne eines solchen Bereichs mit denen anderer Bereiche koordiniert werden.

Die vertikale Koordination beschränkt sich im wesentlichen auf die Vergabe von Fertigungsaufträgen durch eine übergeordnete Dispositionsebene und die Rückmeldungen der Fertigungsinsel an diese Ebene. Zu einer horizontalen Koordination besteht aus zweierlei Gründen Notwendigkeit. Zum einen ist es praktisch nicht auszuschließen, daß Fertigungsaufträge kurzzeitig aus der Insel ausgeschleust werden müssen, um z.B. auf einer inselexternen Spezialmaschine abgearbeitet zu

[39] Vgl. Becker, Rosemann (CIM) (1993), S. 276.
[40] Vgl. Becker, Rosemann (CIM) (1993), S. 277.

werden. Zum anderen kann es durch Störungen innerhalb einer Insel vorkommen, daß abhängige nachfolgende Termine in anderen Fertigungsinseln gefährdet sind.[41]

Leitstandsysteme für die dezentrale Planung in Fertigungsinseln

Die Verlagerung von Tätigkeiten, insb. aus dem Bereich der Arbeitsvorbereitung, in die Fertigungsinsel führt zu einem erweiterten Aufgabenbereich der Mitarbeiter und erweiterter Planungsfreiheit in vielen Bereichen. Durch die hohe Autonomie der Fertigungsinsel haben die Inselmitarbeiter größere Entscheidungsfreiheit in vielen Bereichen. Für die Bewältigung der zusätzlichen Aufgaben und zur Entscheidungsunterstützung bedürfen sie entsprechender Hilfsmittel.[42] PPS-Systeme in Fertigungsinselstrukturen müssen einerseits die Fertigungsinselmitarbeiter bei der Planung unterstützen, und andererseits den entstehenden Koordinations- und Kommunikationsbedarf durch die in Kapitel beschriebenen Möglichkeiten decken. In der Praxis entstand aus diesen Forderungen das Konzept des Fertigungsleitstandes als Bestandteil dezentraler PPS-Systeme.

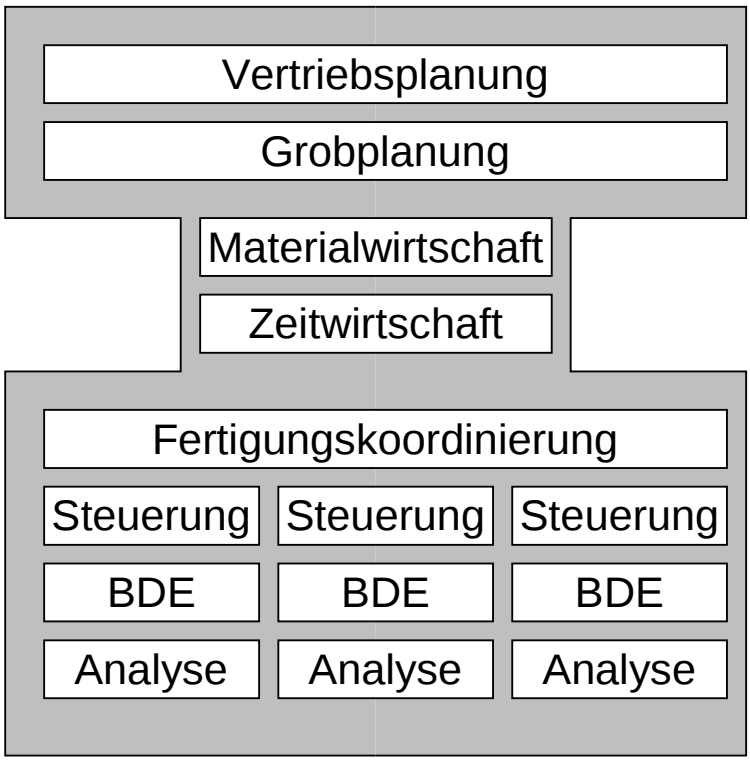
Ursprünglich wurde die Maschinenbelegung in vielen Betrieben auf wandgroßen Tafeln in den Meisterbüros dargestellt. Problematisch bei solchen Plantafeln ist das große Datenvolumen und die unkomfortable Handhabung. Mit den Möglichkeiten der graphischen EDV entstanden als Ersatz die ersten elektronischen Plantafeln. Diese als Fertigungsleitstände bezeichneten Tafeln lassen eine übersichtliche und komfortable Visualisierung des aktuellen Fertigungsgeschehens zu[43]. Ziel der Fertigungsleitstände ist es, im Gegensatz zu klassischen PPS-Systemen, nicht, die Belegungsplanung und den Kapazitätsabgleich mit komplexen Algorithmen selbständig zu erledigen. Vielmehr sollen sie dem Meister oder Werker vor Ort eine sinnvolle und komfortable Planungsunterstützung bieten, indem die Auswirkungen einer Planänderung sofort graphisch und übersichtlich darstellbar sind. Sie übernehmen dabei das Konzept der Plantafel und stellen z.B. die aktuelle Maschinenbelegung als Gantt-Diagramm dar, wobei auf den Diagrammachsen jeweils Betriebsmittel und Zeitverlauf angegeben sind[44]. Sie bieten die Möglichkeit die Kapazitätsbelastung für einzelne Betriebsmittel als sog. Kapazitätsgebirge im Zeitablauf wiederzugeben. Fertigungsauftragspositionen lassen sich graphisch interaktiv verschieben. Mit der Zeit wurde immer mehr Planungs- und Steuerungsfunktionalität in die Fertigungsleitstände übertragen. Sie ergänzen die PPS-Systeme in die fertigungsnahen Bereiche und übernehmen vielfach die Aufgaben,

41 Vgl. Becker, Rosemann (CIM) (1993), S. 280ff.
42 Vgl. Maßberg (1993), S. 35.
43 Vgl. Kurbel (1993), S. 235ff.
44 Vgl. Kurbel (1993), S. 248ff.

die von den klassischen PPS-Systemen nur sehr schwach unterstützt werden.[45] Die Planungen, die mit den Fertigungsleitständen durchgeführt werden, sind näher an den eigentlichen Gegenständen der Planung und können somit schneller und besser auf neue Gegebenheiten reagieren. Fertigungsleitstände vermeiden damit viele der Unzulänglichkeiten klassischer PPS-Systeme.[46]

Die in der Praxis eingesetzten dezentralen Produktionsplanungssysteme bestehen aus einem zentralen PPS-System mit untergeordneten Fertigungsleitständen für die dezentralen Fertigungseinheiten. Der Begriff des PPS-Systems soll im folgenden nur diesen zentralen Teil des gesamten Systems bezeichnen. Im folgenden wird auch der Fertigungsleitstand abgekürzt als Leitstand bezeichnet, obwohl noch weitere Formen denkbar und auch in der Praxis realisiert sind, z.B. Logistikleitstand, Montageleitstand etc. Während das PPS-System weiterhin die klassischen Planungsaufgaben wahrnimmt, werden alle Steuerungsaufgaben in die Leitstände verlagert. Eine sinnvolle Trennstelle ist die Auftragsfreigabe als Übergang von der Produktionsplanung zur Produktionssteuerung. Das PPS-System erzeugt dann für die dezentralen Leitstände einen Auftragsvorrat, dessen Ausführung sie zu steuern haben.

45 Vgl. Scheer, Loos (1995).
46 Vgl. Kurbel (1993), S. 235 ff.



Quelle: Scheer (1990), S. 83.

Abb. 1.5: Gestaltung moderner PPS-Systeme und deren funktionale Abdeckung

Während eine Verteilung von PPS Systemen aus informationstechnischen Gründen, z.B. Lastverteilung, Ausfallsicherheit durch Redundanz, etc. verfolgt wird, erfolgt die Dezentralisierung nach organisatorischen und planerischen Gesichtspunkten. Verteilte PPS-Systeme verlagern Funktionen auf Rechnerknoten die über ein Netzwerk gekoppelt sind. Jeder Knoten hat Zugriff auf die benötigten Daten oder Dienste anderer Knoten, so daß diese Verteilung im allgemeinen für den Benutzer transparent ist. Dezentrale PPS-Systeme sind zwar in dem Sinne verteilt, daß sie die Funktionen auch auf verschiedene Rechnerknoten verlagern, die Verlagerung soll aber für den Benutzer nicht transparent sein. Sie müssen vielmehr in der Lage sein, möglichst vollständig die von einer Organisationseinheit benötigte Funktionalität auf allen Rechnerknoten anzubieten, und dabei individuell auf die Bedarfe der zugeordneten Organisationseinheit eingehen.[47]

Neben der graphischen Darstellungsmöglichkeit und den z.T. umfangreichen Simulationsmöglichkeiten zeichnen sich Leitstandsysteme durch das Vorhandensein weiterer Funktionen für die Planung und Steuerung der Fertigung aus.

[47] Vgl. Kemmer (1993).

Funktionen eines Leitstandes

Die Funktionen, die Fertigungsleitstände übernehmen, lassen sich in die Bereiche der Planung, der Überwachung und der Steuerung aufteilen.[48] Zu den Funktionen der *Planung* gehören die

- Auftragsübernahme
- Die Aufträge werden vom Grobplanungssystem mit geforderten Mengen, benötigten Arbeitsplänen und einzuhaltenden Eckterminen an die Fertigungsinsel übergeben. Arbeitspläne und andere benötigte Stammdaten wie z.B. Stücklisten können entweder vom Grobplanungssystem zentral verwaltet werden oder inselintern in den Leitständen geführt werden.[49]
- Maschinenbelegungsplanung
- Aus den vorhandenen Arbeitsplänen und Aufträgen muß unter Berücksichtigung der in der Insel vorhandenen Ressourcen ein Maschinenbelegungsplan erstellt werden. Dabei sind nicht nur die Maschinen selbst von Interesse. Bei modernen Bearbeitungszentren stehen immer mehr auch Fertigungshilfsmittel als kritische Ressourcen im Vordergrund, die einer Verfügbarkeitsprüfung unterzogen werden und kapazitiv geplant werden müssen.[50] Für die Belegungsplanung sind im wesentlichen alle bisher in klassischen PPS-Systemen zu findenden Maßnahmen anzuwenden, also Losbildung, Splitten, Raffen, etc.[51]
- Auftragsfreigabe
- Innerhalb der Fertigungsinsel umfasst die Auftragsfreigabe das Freigeben der Arbeitsgänge als auch die Materialverfügbarkeitsprüfung sowie die Verfügbarkeitsprüfung für die benötigten Ressourcen. Daneben ist auch die Belegerstellung mit der Auftragsfreigabe verbunden.[52]
- Auftragsrückmeldung

Die *Überwachungsfunktion* des Leitstandes wird durch die Integration von Komponenten der Betriebsdatenerfassung (BDE) übernommen. Dazu gehören die[53]

---

48 Vgl. Abeln (1993), S. 333.
49 Vgl. Ruffing (1991), S. 80.
50 Vgl. Milberg (1993), S. 123.; Moser (1995), S. 72.
51 Vgl. Ruffing (1991), S. 69.
52 Vgl. Kurbel (1993), s. 259.
53 Vgl. Abeln (1993), S. 335.

- Erfassung und Auswertung von Betriebszeiten, Rüstzeiten, Stillstandszeiten der Maschinen,
- Personal- und Zeitdatenerfassung der Mitarbeiter sowie die
- Erfassung der aktuellen Arbeitsgänge an den Maschinen.

Aufgrund der in Abb. 1.4 aufgezeigten Autonomiegrade in vielen weiteren Bereichen neben der dispositiven Autonomie, sollen Leitstände der Fertigungsinsel auch in diesen anderen Bereichen Unterstützung anbieten. Bezüglich der *Steuerung* bieten Leitstände Funktionen für den *DNC-Betrieb* der Fertigungsmaschinen.[54] Die Funktionen des DNC, nach Definition durch die VDI-Richtlinie 3424 als System zur Rechnerdirektführung mehrerer NC-Maschinen, teilen sich in Grund- und Zusatzfunktionen. Die Grundfunktionen umfassen die NC-Programmverwaltung mit den Funktionen der

- Übernahme und Speicherung,
- Bereitstellung,
- Aktualisierung und
- Änderung des Status

von NC-Programmen im System. Die zweite Grundfunktion ist die NC-Datenverteilung. Darunter werden die Teilfunktionen der

- Datenübertragung zu den NC-Steuerungen,
- Rückübertragung von evtl. optimierten NC-Programmen von der Steuerung,
- Übertragung der Werkzeug-Soll-Daten zum Werkzeugeinstellgerät,
- Übertragung der Werkzeug-Ist-Daten vom Werkzeugeinstellgerät,
- Führung der letzten Übertragungsvorgänge für jede Steuerung

verstanden. Die Zusatzfunktionen des DNC umfassen u.a. die NC-Programmerstellung und die zentrale Ausführung von NC-Programmen.

Leitstände übernehmen oft auch Lagerverwaltungsfunktionen, nicht nur für Werkstücke, sondern auch für Fertigungshilfsmittel. Aufgrund deren großer Bedeutung in der flexiblen Automatisierung müssen die Leitstände auch eine Verwaltung bzw.

---

54 Vgl. zu den Ausführungen zum DNC Milberg (1993), S. 116f.

Planung von Betriebsmitteln ermöglichen, die zentrale PPS-Systeme nicht berücksichtigen. Hierzu zählen insbesondere die inselinternen Werkzeuge und Vorrichtungen.[55]

Funktionen des zentralen PPS-Systems

Aufgrund der Verlagerung von Funktionen in die dezentralen Leitstände, ändern sich Art und Umfang der im PPS-System verbleibenden Funktionen. Die auf dieser Ebene benötigten Funktionalitäten sind dann die[56]

- Terminplanung
- Wenn die Arbeitspläne im zentralen PPS-System verwaltet werden, kann durch dieses eine Terminplanung aufgrund der in den Arbeitsplanpositionen vorhandenen Zeiten durchgeführt werden. Wenn nur Rumpfarbeitspläne verbleiben, oder aber die gesamte Arbeitsplanverwaltung in den Leitstand verlagert wurde, muß mit durchschnittlichen Durchlaufzeiten für Teile, evtl. mit geometrieabhängigen Zu- oder Abschlägen, gerechnet werden[57]. Die üblichen hohen Sicherheitspufferzeiten können stark verringert werden, denn sie werden auf dieser Planungsebene nicht benötigt.[58]
- Kapazitätsbedarfsermittlung
- Die Kapazitätsbetrachtung kann auf die Fertigungsinsel als kleinste Kapazitätseinheit beschränkt werden. Wenn zumindest Rumpfarbeitspläne vorhanden sind, können besonders kritische Betriebsmittel der Fertigungsinsel im PPS-System abgebildet und disponiert werden. Fertigungshilfsmittel können entweder im zentralen PPS-System oder auf Leitstandsebene verwaltet werden. Da die meisten in der Praxis eingesetzten PPS-Systeme aber keine Dispositionsunterstützung für Fertigungshilfsmittel bieten, findet sich diese Funktionalität zumeist in den Leitständen.[59]
- Koordination und Steuerung
- Da es trotz hoher Autonomie möglich ist, daß ein Fertigungsauftrag eine Fertigungsinsel zwischenzeitlich verlassen muß, ist eine übergeordnete Koordination durch das PPS-System notwendig. Dazu muß das System in der

55 Vgl. Milberg (1993), s. 123.
56 Vgl. Ruffing (1991), S. 66.
57 Vgl. Ruffing (1991), S. 80.
58 Vgl. Ferstl, Mannmeusel (1995), S. 29.
59 Vgl. Ruffing (1991), S. 81f.

Lage sein, Arbeitsfolgen als Teil eines Arbeitsplanes verwalten zu können. Eine Arbeitsfolge stellt einen durchgängig in einer Fertigungsinsel bearbeitbaren Teil eines Arbeitsplanes dar.[60]

- Einlasten der Aufträge in die Fertigungsinsel
- Das Grobplanungssystem plant Aufträge aufgrund von bekannten Durchlauf- oder Wiederbeschaffungszeiten ein und betrachtet dabei nur die auf Fertigungsinselebene aggregierte Kapazität für die Erzeugung der Auftragsecktermine.
- Entgegennahme von Rückmeldungen der Fertigungsinsel und Terminüberwachung

Verbindung zwischen zentralem PPS-System und Leitständen

Eine Verbindung von PPS-Systemen und Leitständen ist grundsätzlich auf zwei Arten vorstellbar. Bei einer direkten Verbindung des PPS-Systems mit dem Leitstandsystem übernimmt das PPS-System die zentrale Koordination mehrerer Leitstände. Dazu müssen alle Informationen im zentralen PPS-System verfügbar sein, auch wenn sie für die eigentlichen Aufgaben des Systems nicht benötigt werden. So müssen z.B. Rückmeldungen von Auftragspositionen im Gegensatz zu ganzen Aufträgen verarbeitet werden. Die heute eingesetzten PPS-Systeme sind nicht in der Lage, diese Koordination dezentraler Subeinheiten differenziert und zeitnah zu bieten, denn bei den eingesetzten Systemen handelt es sich vielfach um klassische PPS-Systeme die nach unten um Fertigungsleitstände ergänzt wurden. Diese kennen die Begriffe der Fertigungsinsel und Fertigungsinselkapazität nicht.[61] Sie sind ferner nicht in der Lage, mit anderen, auf der gleichen organisatorischen Ebene angesiedelten, Leitständen eine horizontale Kommunikation und Koordination durchzuführen. Zwar existieren erste konkrete Ansätze im Forschungsbereich[62], doch die heute verfügbaren Systeme gehen immer noch von rein vertikaler Kommunikation mit einer top-down-Detailplanung aus.

Die Anpassung solcher Systeme bzw. deren Ersatz ist ökonomisch kaum vertretbar.[63] Eine andere Möglichkeit der Verbindung ist daher das Einfügen einer speziellen Koordinationsinstanz zwischen dem PPS-System und den Leitständen, die genau diese Funktionalität bietet. Diese Instanz kann entweder auf der gleichen Hierarchieebene wie die Leitstände eingefügt sein oder hierarchisch übergeordnet sein.[64]

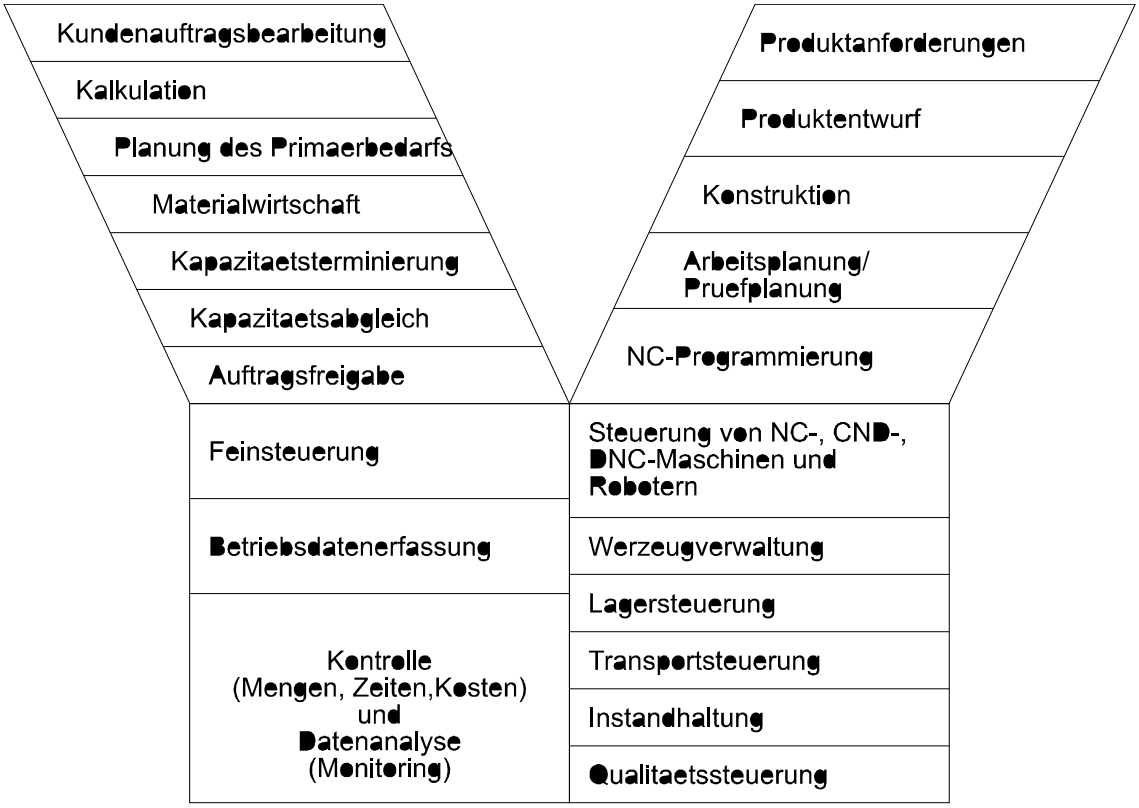
---

60 Vgl. Ruffing (1991), S. 82.
61 Vgl. Kurbel (1993), s. 262.; Maßberg (1993), S. 50.
62 Vgl. Schneider, Rinschede (1996), S. 52-61.
63 Vgl. Maßberg (1993), S. 51.
64 Vgl. Kurbel (1993), S. 261ff.

Integration dezentraler PPS-Komponenten durch Schnittstellensysteme

Der Begriff des Computer Integrated Manufacturing (CIM) beschreibt die „integrative Gesamtsicht auf sämtliche betriebswirtschaftlich-dispositiven und technischen Aufgaben einer Unternehmung“[65] und befaßt sich mit der integrativen Betrachtung des Informationsflusses und der Informationsverarbeitung. Neben der technischen Seite hat das CIM auch eine organisatorische Komponente, denn erst durch die Analyse und das Verständnis der betrieblichen Abläufe läßt sich die Voraussetzung für die CIM-Realisierung schafffen.[66]

Der Begriff des CIM umfaßte ursprünglich nur den heute als CAM bekannten Bereich der technischen Funktionen der Fertigung, wurde um die Funktionen des CAD, CAP und CAQ erweitert und umfaßt heute auch die Aufgaben der Auftragsabwicklung und der Beschaffung.[67] Dieses wird am CIM-Y-Modell aus Scheer (1990) deutlich:



Quelle: Angelehnt an Scheer (1990), S. 2.

Abb. 1.1: CIM-Y nach Scheer

Dieses Modell betont die gleichgewichtige Stellung der betriebswirtschaftlichen und der technischen Funktionen. Es zeigt die zeitliche Fristigkeit der Funktionen, die sich in

65 Becker, Rosemann (CIM) (1993), S. 14.
66 Vgl. Abeln (1993), S. 439.
67 Vgl. Becker (1991), S. 12f.

eine Planungsteil, die beiden Schenkel des Y, und den Steuerungsteil, den Stamm des Y, teilen lassen. Andererseits wird deutlich, daß sich die Funktionen der linken Seite mit dem Objekt des Auftrags befassen, die der rechten Seite das Produkt als zentrales Objekt betrachten.[68] Das Zusammenkommen der betriebswirtschaftlichen und der technischen Funktionen bei kürzer werdendem zeitlichen Betrachtungshorizont und schließlich die enge Verzahnung der beiden bei der Produktionssteuerung bzw. Fertigung wird durch dieses Modell sehr deutlich dargestellt.

Die Integration des Informationsflußes und der Informationsverwaltung ist aber nicht Selbstzweck, sondern verfolgt konkrete betriebswirtschaftliche Ziele. Diese haben sowohl direkte als auch indirekte Kostenwirkungen. Direkte Kostensenkungen ergeben sich, wenn Funktionen auf den Datenbestand anderer Systeme zurückgreifen können. Dies macht die Mehrfacheingabe von Daten überflüssig, die einerseits sehr langsam und zeitaufwendig ist, andererseits eine Fehlerquelle darstellt. Fehler können unbeabsichtigt falsche Eingaben sein oder inkonsistente Daten in zwei Funktionsbereichen.[69]

Ein weiteres Ziel von CIM ist die Reduzierung der Zahl der Schnittstellen durch das Zusammenfassen von Funktionen, da jede Schnittstelle den Informationsfluß verlangsamt und eine potentielle Fehlerquelle darstellen kann. Durch schnelleren und besseren Informationsfluß verkürzt sich die Durchlaufzeit der Aufträge. Dadurch kann eine höhere Termintreue erreicht werden. Niedrige Durchlaufzeiten erhöhen wiederum die Flexibilität der Fertigung bezügl. Produktänderungen. Dadurch werden immer kleinere Losgrößen wirtschaftlich.

Arten der Integration

Die Integration der betrieblichen Bereiche kann nach folgenden Gesichtspunkten systematisiert werden. Dabei bilden die Daten- und Datenstrukturintegration die Voraussetzung für die Funktions- und Modulintegration.

Datenintegration

Datenintegration bedeutet die gemeinsame Nutzung derselben Daten durch unterschiedliche Unternehmensbereiche. Der Wegfall von Mehrfacheingaben bringt eine starke Verringerung des Änderungsaufwandes mit sich, da die Daten nur einmal eingegeben, bzw. geändert werden müssen. Durch die einmalige Speicherung der Daten werden Inkonsistenzen zwischen den unterschiedlichen Datenbeständen und bei der

68 Vgl. Becker (1991), S. 7.
69 Vgl. Becker (1991), S. 166.

Informationsübermittlung vermieden, da alle Bereiche auf denselben Datenbestand zugreifen. Datenintegration bedeutet damit auch eine schnellere Informationsübertragung, im Idealfall die sofortige Verfügbarkeit von Daten. Änderungen, die in einem betrieblichen Bereich vorgenommen werden, sind sofort für alle anderen Bereiche sichtbar.[70]

Datenstrukturintegration

Eine Ausprägung der Datenstrukturintegration ist die Wiederverwendung der Struktur eines einzelnen Datensatzes. So können Werkzeuge und Vorrichtungen zu Fertigungshilfsmitteln zusammengefaßt werden, da sich ihre Stammsätze im Aufbau nur geringfügig unterscheiden. Die zweite Ausprägung der Datenstrukturintegration besteht in der Wiederverwendung von Beziehungen zwischen Datensätzen. So kann z.B. die Beziehung der Stücklistenstruktur sowohl für Werkzeuge als auch Vorrichtungen verwendet werden.[71]

Modulintegration

Unter Modulintegration versteht man die gemeinsame Benutzung von EDV-Modulen durch mehrere Unternehmensbereiche. Ein Beispiel ist die Lagerhaltung, die für Werkzeuge (Werkzeugabteilung), Fertigprodukte (Verkauf), Rohteile (Einkauf), Halbfertigware (Fertigung) etc. benutzt werden kann.[72] Modulintegration kann sich auf zwei Arten vollziehen. Zum einen kann ein Modul von mehreren betrieblichen Bereichen benutzt werden. Dies ist insb. der Fall, wenn die Bereiche mit einem einzigen Rechner arbeiten. Im anderen Fall wird das gleiche Programm auf mehreren Rechnern eingesetzt. Voraussetzung hierfür ist allerdings sowohl Hardware- als auch Betriebssystem- und Datenbankkompatibilität. Hier setzen sich in jüngster Zeit immer mehr UNIX als Betriebssystem und relationale Datenbankmanagementsysteme (DBMS) mit SQL (structured query language) als DDL, DML und Abfragesprache durch. Darüber hinaus müssen neben der Datenbank auch die gleichen Datenstrukturen auf beiden Systemen festgelegt sein. Notwendig für die Modulintegration ist also die Datenstrukturintegration.[73]

---

70 Vgl. Becker (1991), S. 166.
71 Vgl. Becker, Rosemann (CIM) (1993), S. 16f.
72 Vgl. Becker, Rosemann (CIM) (1993), S. 18.
73 Vgl. Becker (1991), S. 179.

Funktionsintegration

Auch die Funktionsintegration kennt zwei Ausprägungen. Zum einen wird Integration hier verstanden als das Zusammenwachsen von vormals getrennten Funktionen. Dieses Verschmelzen ist immer dann sinnvoll, wenn dadurch vormals sequentiell ablaufende Funktionen vereinigt werden können, so daß Übergangszeiten verringert werden können. [74] Funktionsintegration bedeutet auch die Möglichkeit des Auslösens von Funktionen durch andere Funktionen. Diese als Triggern bezeichnete Auslösung bewirkt eine Verkürzung der Liegezeit zwischen Funktionen. Da diese Zeit einen bedeutenden Anteil an der Durchlaufzeit ausmacht, ergeben sich hier große Rationalisierungspotentiale.[75]

Möglichkeiten der Integration

Schwierigkeiten bei der Systemintegration werden insbesondere durch Daten hervorgerufen, die in den verschiedenen Systemen unterschiedliche Werte annehmen oder auf die eine unterschiedliche Sicht, z.B. Fertigungs- vs. Konstruktionsstückliste, existiert. Grundsätzlich existieren verschiedene Möglichkeiten der Datenintegration, die auch als CIM-Integrationsstufen bezeichnet werden. Im folgenden wird die Klassifizierung nach Becker (1991) übernommen. Darin lassen sich auch die 5 Stufen nach Ruland (1991) [76] einordnen.

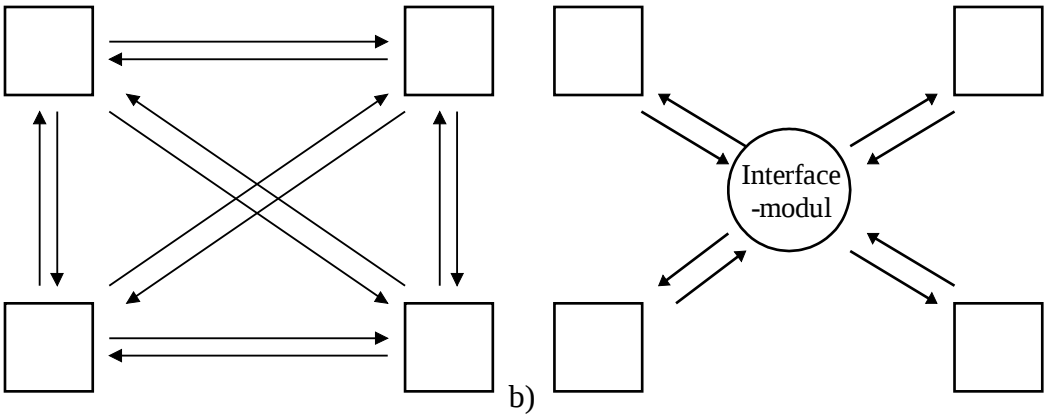
Direkte Kopplung von Systemen

In der einfachsten Form der Kopplung, *der Integrationsstufe 0*, besteht zwischen den Systemen kein Kommunikationssystem, so daß die Daten manuell neuerfaßt werden müssen.[77]

Als nächsthöhere Form der Integration besitzen die Systeme in der *Integrationsstufe 1* die Möglichkeit, gegenseitig Daten auszutauschen. Dabei kann prinzipiell jedes System mit jedem anderen in beiden Richtungen Daten austauschen, evtl. mittels Koppelprozessoren. Diese übernehmen sowohl die syntaktische als auch die semantische Transformation der auszutauschenden Daten zwischen den beteiligten Sytemen[78]. Die direkte Kopplung stellt keine Integration im eigentlichen Sinne dar, da

74 Vgl. Becker (1991), S. 191.
75 Vgl. Becker (1991), S. 183.
76 Vgl. Ruland (1991), S. 109ff.
77 Vgl. Ruland (1991), S. 111.
78 Vgl. Ruland (1991), S. 112.

die Systeme immer noch eine eigene Datenhaltung haben und so Möglichkeiten für das Auftreten von Inkonsistenzen vorhanden sind.[79]



Quelle: Angelehnt an Becker, Rosemann (CIM) (1993), S. 21.

Abb. 1.2: Zahl der Schnittstellen bei direkter und indirekter Systemkopplung

Bei dieser Art der Integration wächst die Anzahl der Schnittstellen zwischen den Systemen quadratisch mit der Anzahl der eingesetzten Systeme an (vgl. Abb. 1.2 a). Dieses bedeutet sehr viel Entwicklungs- bzw. Pflegeaufwand, sollten sich die Datenformate im Laufe der Zeit ändern.[80] Problematisch bei der direkten Kopplung sind weniger die unterschiedlichen Datenstrukturen, die sich EDV-technisch umsetzen lassen. Vielmehr sind unterschiedliche Inhalte oder unterschiedliche Sichten der beteiligten Systeme auf die Daten die eigentlichen Probleme, die oftmals nur durch interaktives Eingreifen des Benutzers lösbar sind.[81] Dieses Problem der sog. Differenzdaten und Datenentropie wird in der *Integrationsstufe 2* durch ein gemeinsames Referenzinformationsmodell umgangen, wobei jedoch der Datenaustausch immer noch über Koppelprozessoren abläuft.[82]

Durch die direkte Kopplung von Systemen läßt sich die Art der Funktionsintegration realisieren, die als Triggern bezeichnet wird. Wenn die beteiligten Systeme in der Lage sind, von anderen Systemen Anweisungen zu erhalten, läßt sich so eine beschränkte Funktionsintegration herstellen. Funktionsintegration im Sinne von Zusammenwachsen von Systemen und Modulintegration ist bei dieser Form der Kopplung nicht möglich[83].

---

79 Vgl. Becker (1991), S. 194.
80 Vgl. Ruland (1991), S. 113.
81 Vgl. Ruland (1991), S. 113.
82 Vgl. Ruland (1991), S. 114.
83 Vgl. Becker (1991), S. 194.

Datenübertragung mit Schnittstellensystemen

Eine direkte Kopplung der Systeme ist insbesondere aufgrund der großen Anzahl der Schnittstellen nachteilig. Die nächste Stufe der Integration ist die Möglichkeit, die beteiligten Systeme über ein zentrales Modul zu integrieren. Die Systeme kommunizieren nicht mehr direkt untereinander, sondern tauschen Informationen über dieses zentrale Integrationsmodul aus. Die Daten der Quellsysteme werden dabei in das Referenzinformationsmodell transformiert. Dadurch wird die Anzahl der benötigten Schnittstellen stark reduziert. Sie wächst nur noch linear mit der Anzahl der Systeme (vgl. Abb. 1.2 b). Dieses Konzept wird als *Integrationsstufe 3* bezeichnet.[84]

Insbesondere wenn bereits CIM-Komponenten vorhanden sind, wie in vielen Betrieben der Fall, erweist sich diese Möglichkeit der Integration als vorteilhaft, denn bestehende Systeme sind oft nicht in der Lage, mehr als eine Schnittstelle zur Verfügung zu stellen. Sie eignen sich damit nicht für die direkte Kommunikation mit mehreren anderen Systemen. Es ist aber oftmals technisch nicht möglich, die Systeme dahingehend zu verändern, daß diese Möglichkeit hergestellt wird, noch ist es wirtschaftlich vertretbar, die eingesetzten Systeme aus diesem Grund durch offenere Systeme zu ersetzen.[85]

Durch die geringere Anzahl der Schnittstellen wird der evtl. Änderungsaufwand stark reduziert. Nachteilig ist aber der hohe Entwicklungs- und Realisierungaufwand, denn ein solches Integrationsmodul muß neben der eigentlichen Aufgabe der Datenübertragung diese Daten meist noch transformieren, bzw. den Informationsfluß zeitlich steuern. Somit müssen die Schnittstellen selbst als „intelligente“ Module ausgelegt sein, die die Konsistenz und Integrität der Daten (Typ- und Schlüsselintegrität sowie die semantische Integrität) über mehrere Systeme weitestgehend sicherstellen.

Gemeinsame Datenbasis

Eine gemeinsame Datenbasis durch den Einsatz moderner Datenbankmanagementsysteme (DBMS) bietet die Vorteile, daß hierdurch nicht nur die Datenintegration sondern auch die Datenstrukturintegration auf optimale Weise möglich ist. Voraussetzung für diese *4. Integrationsstufe* ist der Aufbau eines Datenmodells, z.B. als ER-Modell, für alle betroffenen Untenehmensbereiche und die Realisierung dieses unternehmensweiten Datenmodells (UDM) in einer Datenbank. [86] Diese Datenbank kann von allen Unternehmensbereichen und Funktionen genutzt werden. Daten werden hierin redundanzfrei dargestellt. Moderne DBMS stellen auch

---

84 Vgl. Ruland (1991), S. 114f.
85 Vgl. Maßberg (1993)
86 Vgl. Ruland (1991), S. 116f.; Becker (1991), S. 194ff.

Möglichkeiten zur Integritäts- und Konsistenzsicherung (referentielle Integrität, Typintegrität, etc.) zur Verfügung und ermöglichen unterschiedliche Sichten auf die Daten (z.B. View-Konzepte bei relationalen Datenbanken). Nachteilig ist aber insb., daß sich dieses Konzept aufgrund der Beschränkung vorhandener Systeme vielfach nicht in bestehende Systemstrukturen einfügen läßt.[87]

Realisierungs-möglichkeit Integrations-komponente	direkte Kopplung	CIM-Interface-System	gemeinsame Datenbasis
Datenintegration	Abgleich redundanter Datenbestände über direkte Kopplung	Abgleich redundanter Datenbestände über das CIM-Interface-System	logisch redundanzfreier Datenbestand
Datenstruktur-integration	nein	indirekt, wenn das CIM-Interface-System die Schnittstellenstrukturen vorgibt	gut
Modul-integration	nein	nein	indirekt über Daten- und Datenstrukturintegration
Funktions-integration a) Vereinigung	nicht ohne Eingriff in bestehende Systeme	indirekt über Daten- und Datenstrukturintegration	nicht ohne Eingriff in bestehende Systeme
Funktions-integration b) Triggern	nein, evtl. über Programm-Programm-Kommunikation		nein, evtl. über Programm-Programm-Kommunikation

[87] Vgl. Ruland (1991), S. 118.

Quelle: Becker, Rosemann (CIM) (1993), S. 22.

Tab. 1.1: CIM-Integrationsalternativen

Aufgaben von Schnittstellensystemen

Ein Schnittstellensystem im Sinne des CIM muß neben der Übertragung von Daten auch die Integrität der Daten gewährleisten, diese bei Bedarf tranformieren und synchronisieren.

Datenintegrität

Das Schnittstellensystem muß verschiedene Arten der Datenintegrität zwischen den beteiligten Systemen wahren. Diese Arten lassen sich unterscheiden, ob sie statischer oder nicht-statischer Natur sind.[88] Zu den *statischen Integritätsbedingungen* zählen die

- Datenfeld-Bedingungen (Attributbedingungen) sind Bedingungen, die die Werte eines einzelnen Datums einschränken
    - Vorhandensein des Datums
    - Typbedingungen (legen den Datentyp des Datums fest)
    - Unter- / Obergrenzen (es können bestimmte Grenzen für die Werte des Datums, insb. bei numerischen Datentypen, festgelegt werden)
    - Bestimmte Werte (die möglichen Werte des Datums werden explizit durch Enumerierung angegeben)
- Datensatzbedingungen (Tupelbedingungen) schränken den Wertebereich eines Datensatzes ein, indem sie die Datenfelder eines Datensatzes miteinander in Verbindung bringen

  Bsp.: Die Ist-Ausbringung ist kleiner oder gleich der Sollmenge

- Datei-, Tabellen-, Relationenbedingungen schränken den Wertebereich aller Datensätze einer Tabelle/Datei ein
    - Schlüsselbedingungen (geben an, daß ein bestimmtes Datum ein Schlüsseldatum sei, daß also Eindeutigkeit in Bezug auf dieses Datum besteht)

[88] Vgl. zu den folgenden Ausführungen Vossen (1993), S. 512ff.

- Aggregatbedingungen (geben an, wie sich eine Aggregierung eines Datums über mehrere Datensätze verhalten muß)

  Bsp.: Die Summe aller nachgefragten Kapazitätseinheiten muß kleiner sein als eine bestimmte Obergrenze

- Rekursive Bedingungen

  Bsp.: Eine Maschine muß von jeder Bearbeitungsart zu jeder anderen Bearbeitungsart umrüstbar sein.

- Referentielle Bedingungen

  Bsp.: Ein Teil darf nur in einer Stückliste referenziert werden, wenn es einen Teilestammsatz hat

Bei den nicht-statischen Integritätsbedingungen wird unterschieden zwischen den

- *transitionalen Bedingungen,* die die Menge der möglichen Übergänge zwischen zwei gültigen Datenbankzuständen einschränken und

  Bsp.: Die Istmenge des Auftrages darf nur größer werden.

- *dynamischen Bedingungen* als Verallgemeinerung auf Übergangsfolgen zwischen mehr als zwei gültigen Datenbankzuständen.

Integrationsbedingungen lassen sich aber auch in lokale und globale Bedingungen teilen. *Lokale Integritätsbedingungen* lassen sich bei einer interaktiven Dateneingabe durch das jeweilige System überprüfen. Wenn jedoch die Daten durch elektronischen Datenaustausch in das System gelangen sollen, muß geprüft werden, ob eine Verwendung dieser Prüffunktionen noch möglich ist. Ist dieses nicht der Fall, so müssen die lokalen Integritätsbedingungen im Schnittstellensystem überwacht werden. *Globale Integritätsbedingungen* lassen sich alternativ in jedem der angeschlossenen Systeme überprüfen oder aber zentral durch das Schnittstellensystem. Die Prüfung durch das Schnittstellenystem hat den Vorteil, daß die Daten bereits über das Schnittstellensystem übertragen werden müssen und eine weitere Übertragung nur für den Zweck der Integritätsprüfung deshalb entfallen kann.[89]

[89] Vgl. Becker, Priemer (1993), S. 146f.

Datensynchronisation

Neben den Integritätsbedingungen muß die Synchronisation der Datenbanken der Systeme sichergestellt werden. Es können dabei prinzipiell die folgenden Probleme auftreten[90]:

- Ein System ist vorübergehend nicht betriebsbereit.
- Es treten Fehler bei der Datenübertragung auf.
- In mehreren Systemen werden gleichzeitig Änderungen desselben Datums durchgeführt.
- In einigen Systemen ist eine Änderung erfolgreich, in anderen Systemen nicht.

Generell sind diese Probleme durch Transaktionskonzepte wie sie auch bei verteilten Datenbanken eingesetzt werden, zu lösen.[91] Aus Praktikabilitätsgründen sollte jedoch eine Abschwächung der strengen Forderungen des Transaktionskonzepts erfolgen.[92]

Im betrachteten Fall der ABB Kraftwerke AG treten die Probleme der Datensynchronisation jedoch nur in einem sehr geringen Ausmaß auf, denn zum einen verläuft der Informationsfluß unidirektional, zum anderen ist die Zahl der beteiligten System sehr beschränkt. Auf verteilte Transaktionskonzepte kann daher verzichtet werden. Die Anforderungen an die Datenübertragung, insb. die Zeitpunkte der Übertragungsauslösung, werden in Kapitel näher beschrieben.

Datentransformation

Neben der Integritätssicherung und der Synchronisation der Daten müssen diese in den meisten Fällen auch transformiert werden. Diese Transformation vom jeweils eigenen Datenmodell in das Referenzmodell des Interface-Systems wird auch als Modellkopplung bezeichnet. Im einfachsten Fall benutzen unterschiedliche Systeme nur unterschiedliche Einheiten für Datenfelder. Kompliziertere Fälle können z.B. das Berechnen von benötigten Datenfeldern aus mehreren Ausgangsfeldern erfordern.[93]

---

90 Vgl. Becker (1991), S. 209.
91 Vgl. Becker, Priemer (1993), S. 152ff.
92 Vgl. Becker (1991), S. 210.
93 Vgl. Ruland (1993), S. 124ff.

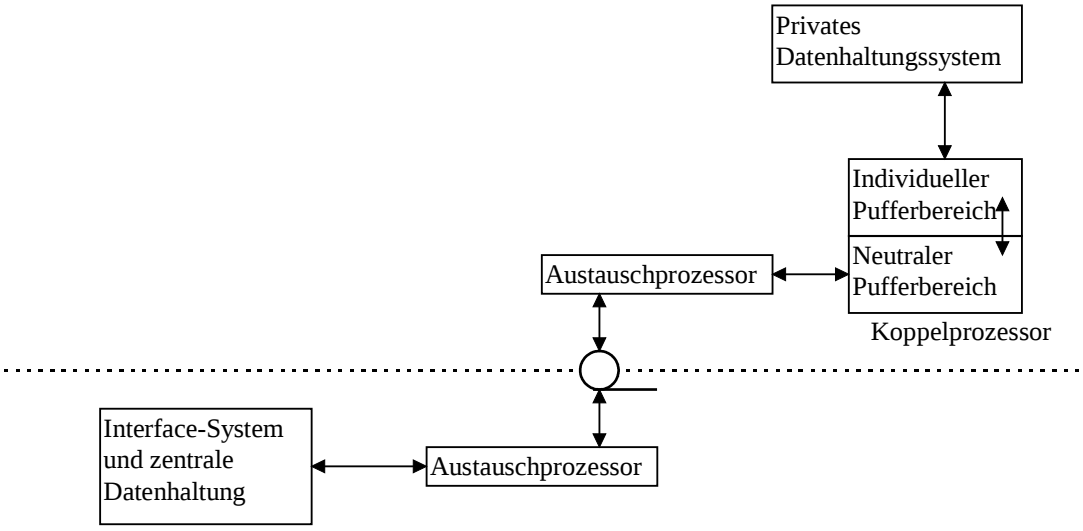
Architektur von Schnittstellensystemen

Da heutige Systeme, insbesondere Standardsoftwaresysteme, auf einer eigenen Datenbasis aufbauen, und somit die Datenstrukturen zwischen den Systemen unterschiedlich sind, ist es im allgemeinen nicht möglich, auf einer gemeinsamen Datenbank mit einem unternehmensweiten Datenmodell (UDM) aufzubauen. Aufgrund der Mängel der direkten Verbindung von Systemen bleibt die Möglichkeit des zentralen Schnittstellensystems. Dieses kann entweder als Bus-System oder als zentrale Instanz ausgelegt sein. Bei einem Bus-System greifen die angeschlossenen Systeme nur die Daten ab, die sie verarbeiten können. Die Daten können daher ohne eine Zielsystemangabe auf den Bus übertragen werden.[94] Ein Bus-System macht jedoch die Klassifikation der über den Bus übertragenen Daten notwendig und erzeugt dadurch einen Overhead für die Kodierung und Dekodierung der Klassifikationsdaten, der das Sytem unproduktiv belastet. Eine andere Möglichkeit eines Schnittstellensystems, welche die Klassifizierung der übertragenen Daten überflüssig macht, ist die Realisierung durch eine zentrale, steuernde Instanz.[95]

Die Hauptkomponten eines Schnittstellensystems sind dann das zentrale Datenhaltungssystem und die Koppelprozessoren. Zum Umfeld zählen die privaten Datenhaltungssysteme der beteiligten Systeme. In einer verteilten Umgebung werden zusätzliche Austauschprozessoren benötigt, die für die Datenübertragung zwischen den Koppelprozessoren zuständig sind. Die Koppelprozessoren übernehmen die semantische und syntaktische Transformation der zu übertragenden Daten. Die Austauschprozessoren überführen die Daten anschließend in das lineare Austauschformat, i.a. eine Datei, die dann übertragen wird.[96]

---

94 Vgl. Balgheim, Jansen (1993), S. 13f.
95 Vgl. Becker, Priemer (1993), S. 145.
96 Vgl. Ruland (1993), s. 129f.



Quelle: Angelehnt an Ruland (1991), S. 129.

Abb. 1.3: Schnittstellensystemarchitektur

Die Austauschprozessoren bestehen jeweils aus dem Compiler, der aus den rechnerinternen Darstellungen die zu übertragenden Datenströme erzeugt, und dem Parser, der die inverse Funktionalität besitzt.[97]

Aus den obigen Ausführungen wird deutlich, daß die Verbindung dezentraler PPS-Komponenten neben den Aufgaben des in besprochenen Koordinationsknotens für mehrere Leitstände und der Informationsflußsteuerung auch die Datentransformation und die Integritätssicherung übernehmen muß. Aufgrund der hohen Komplexität des zu entwickelnden Systems muß ein strukturiertes Vorgehen bei der Systementwicklung gewählt werden. Das Software-Engineering bietet hierzu geeignete Modelle und Methoden an.

[97] Vgl. Ruland (1991), S. 129.

Vorgehensweise bei der Schnittstellensystementwicklung

Mit der zunehmenden Komplexität der Systeme und der zunehmenden Bedeutung dieser Systeme als kritische Erfolgsfaktoren für die Unternehmen muß ihre Erstellung mit ingenieurmäßigen Methoden und automatisierten Hilfsmitteln stattfinden. Es bedarf daher eines ingenieurmäßigen Vorgehens um die bestehende Krise in der Datenverarbeitung, hervorgerufen durch Kosten- und Terminüberschreitungen und unzuverlässige Produkte[98], zu überwinden.[99]

Das wichtigste Prinzip des Software-Engineerings ist die Strukturierung von großen Projekten in besser handhabbare Teile. Eine Möglichkeit besteht darin, den Entwicklungsprozeß in eine Anzahl wohldefinierter Phasen zu unterteilen. Diese Teilung führt zum Konzept der Software Life-Cycle-Modelle. Eine weitere Alternative besteht in der Partitionierung des zu erstellenden Systems in Module als Teil des Systems, welche mit anderen Modulen in Verbindung gebracht werden. Eine dritte Möglichkeit besteht darin, das System in verchiedene Ansichten zu gliedern. Diese Zerlegung ist einer der Ursprünge des ARIS-Modells.[100]

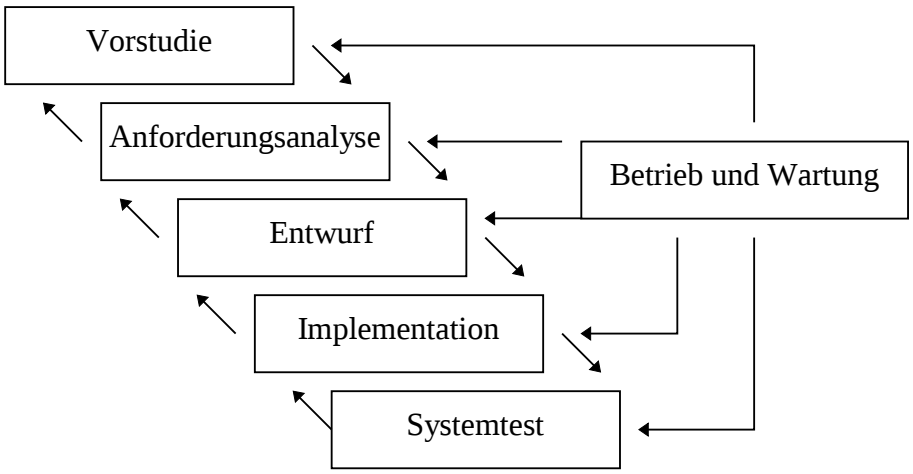
Aufgrund der hohen Komplexität beim Entwurf der von modernen Unternehmen benötigten Anwendungen ist es kaum möglich, dies ohne geeignete Werkzeuge zu versuchen. Der CASE-Begriff (computer aided software engineering) beschreibt die für die rechnergestützte Systemanalyse und das Systemdesign entwickelten Werkzeuge. Diese verwenden präzise diagrammatische Darstellungen, die durch Rechnereinsatz konsistent gehalten werden. Wie in anderen Bereichen der Ingenieurwissenschaften stellen auch im Software-Engineering die Diagramme die Dokumentation des Systems dar. Wenn das System geändert wird, müssen Änderungen an den Diagrammen vorgenommen und der Programmcode daraus neu generiert werden.[101]

Vorgehens- und Life-Cycle-Modelle

Die ersten Software-Lebenszyklus-Modelle sind dadurch gekennzeichnet, daß sie den gesamten Life-Cycle eines Applikationssystems in Phasen unterteilen, die nacheinander durchlaufen werden. Das Ergebnis der vorgehenden Phase ist der Ausgangspunkt für die Arbeiten in der folgenden Phase. Hieraus rührt auch der für diese einfachen Modelle verwendete Begriff des *Wasserfall-Modells*. Nachteilig ist dabei die in der Praxis unvermeidbare Überschneidung der Phasen und die Konzentration von Test und

98 Vgl. Berztiss (1996), S. 45f.
99 Vgl. Martin (1989), S. 5f.
100 Vgl. Berztiss (1996), S. 57f
101 Vgl. Martin (1989), S. 29ff.

Wartung auf das Ende des Prozeßes, wenn Änderungen nur noch mit sehr großem Aufwand möglich sind. Wenn in einer nachfolgenden Stufe Unzulänglichkeiten festgestellt werden, so muß dieses möglichst unmittelbare Auswirkungen auf die vorhergehende Stufe haben. Trotz dieser Mängel kann dieses idealisierte Modell als Referenz verwendet werden.[102]



Quelle: Schönthaler, Nemeth (1990), S. 21

Abb. 2.1: Idealisiertes Life-Cycle-Modell

Dem verwendeten CASE-Werkzeug (Oracle Designer/2000) liegt ein Vorgehensmodell zugrunde, welches sich an dieses idealisierte Modell anlehnt. Im folgenden sollen die Inhalte der einzelnen Phasen kurz dargestellt werden, unter Bezugnahme auf die konkret verwendete Methode, der in BARKER (1990) vorgestellen Oracle CASE*Methode.

In der *Vorstudie* muß die Ausgangssituation mit den bestehenden Systemen untersucht und qualitativ beschrieben werden. Dabei sollen die bestehenden Schwachstellen herausgearbeitet werden. Ausgehend von den ermittelten Schwachstellen müssen dann konkrete Strategien und Ziele für die Entwicklung von Informationssystemen abgeleitet werden.

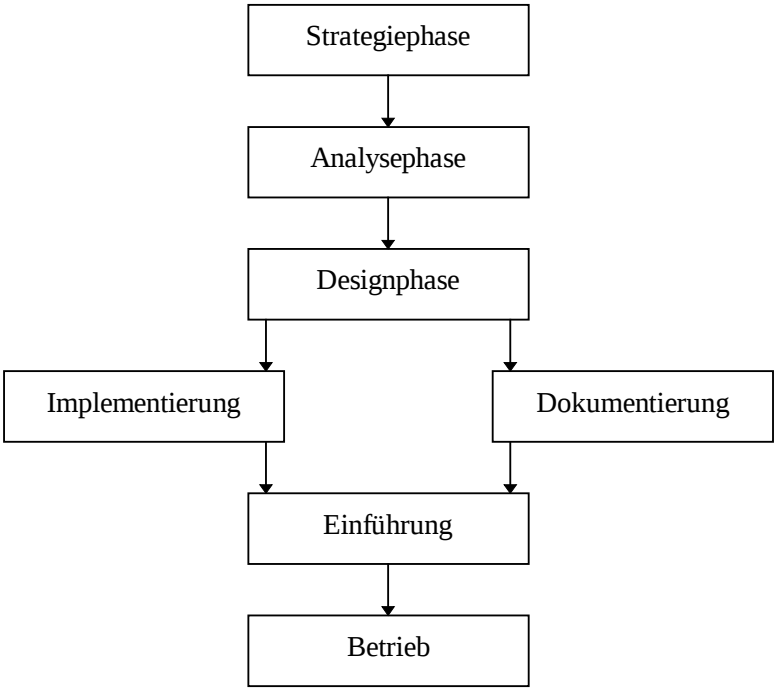
Die Phase der *Anforderungsanalyse* umfaßt die möglichst präzise Spezifikation der fachlichen Anforderungen des zu entwickelnden Systems in einem Fachkonzept. Dieses wird zunächst in natürlicher Sprache ausgedrückt und mit der Zeit um Diagramme ergänzt, die die Anforderungen wiedergeben. In diese Phase fällt z.B. die Erstellung des Daten- und Funktionenmodells.

---

[102] Vgl. zu Life-Cycle Modellen Schönthaler, Nemeth (1990), S. 20ff; Berztiss (1996), S. 83ff. Ein weiteres Vorgehensmodell wird in Becker (HANDEL) (1996), S. 93-122 vorgestellt. Auch dieses läßt sich in das beschriebene Modell einordnen.

Die *Entwurfsphase* umfaßt die Abbildung der Funktionen- und Datenstrukturen des Fachkonzepts in Formen, die eine konkrete Implementierung ermöglichen. Hier werden u.a. ER-Modelle in ein Datenbankdesign überführt und Bildschirmmasken für die Funktionen festgelegt.

Während der *Implementierungsphase* werden die Entfwürfe durch Programmodule und Datenbankbeschreibungen realisiert. Die Module werden mit verschiedenen Hilfsmitteln programmiert und die Datenbank mit DDL (data definition language)-Befehlen in einem konkreten DBMS eingerichtet. Daneben umfaßt diese Phase auch den Test der einzelnen Module und des ganzen Systems. Hier weicht die Oracle CASE*Methode vom idealisierten Modell ab, es lassen sich aber dennoch alle Aktivitäten in beiden Modellen wiederfinden. Während der Systemtest im idealisierten Modell eine Entwicklungsphase bildet, ist er bei der CASE*Methode Bestandteil der Implementierungsphase. Umgekehrt kann die Dokumentationsphase der Implementierungsphase im idealisierten Modell zugeordnet werden. Die Einführungsphase umfaßt neben der Schulung der Benutzer auch den Abnahmetest. Hardware und Software werden installiert und eventuelle Altdaten in das neue System übernommen und aufbereitet. Diese Aktivitäten finden sich im idealisierten Modell bereits in der Phase des Betriebs, die unmittelbar nach Installation und Übernahme des Systems in den laufenden Betrieb beginnt und neben der Wartung des Systems und evtl. Fehlerbeseitigung auch notwendige Änderungen umfaßt.

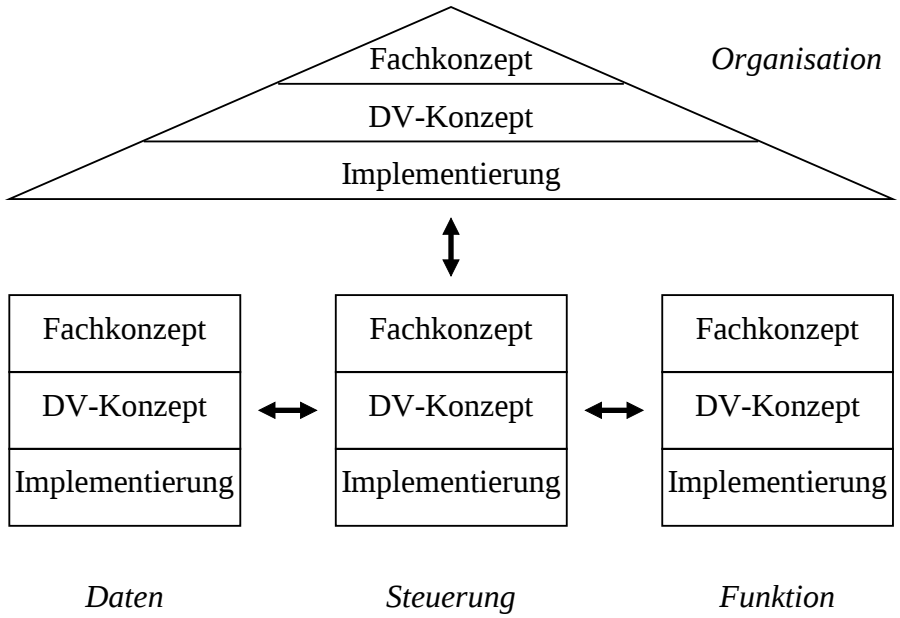


Quelle: Barker (1990), S. 1-3

Abb. 2.2: Life-Cycle-Modell der CASE*Methode

Eine weitere Möglichkeit, die Komplexität von Systemen zu reduzieren, besteht in der Bildung von *Sichten* auf das System. Dadurch besteht die Möglichkeit, einzelne Sichten durch besondere Methoden zu beschreiben, die in ein Gesamtmodell integriert werden. Im *ARIS-Modell* (Architektur integrierter Informationssysteme) sind dies die Organisationssicht, die Datensicht, die Funktionssicht und deren Integration in der Steuerungssicht. Innerhalb der Sichten orientiert sich die ARIS-Architektur an den Life-Cycle- Konzepten zum Software-Engineering, die hier aber kein Vorgehensmodell darstellen, sondern drei Schichten in unterschiedlicher Nähe zur Informationstechnik definieren. Diese Schichten leiten sich aus den Phasen Anforderungsanalyse, Entwurf und Implementation ab.[103]

[103] Vgl. Scheer (1994), S. 10-16.



Quelle: Scheer (1994) S. 17.

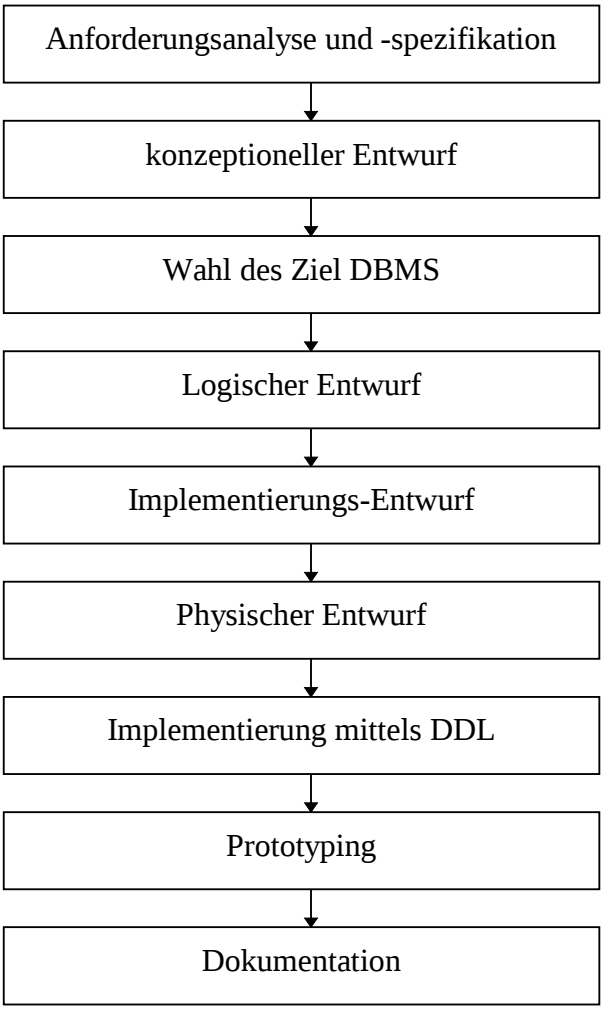
Abb. 2.3: ARIS-Konzept

Beim Entwurf eines Schnittstellensystems stehen die Daten und deren Integrität, Transformation und Übertragung im Vordergrund. Die Datensicht muß daher besondere Beachtung finden.

Datenbankentwicklungsprozeß

Informationsanforderungen müssen unabhängig von möglichen Speicherungs- oder Zugriffsmethoden definiert werden.[104] Ferner sind Informationssysteme nach einer Datenmodellierung sehr viel einfacher zu erstellen; die Erstellung von Systemen ohne Datenmodelle ist kaum vorstellbar.[105] Das hier vorgestellte, aus VOSSEN (1994) entnommene Modell lehnt sich an das idealisierte Life-Cycle-Modell an. Jede der Phasen des Life-Cycle-Modells findet in dem folgenden Modell eine Entsprechung.[106]

104 Vgl. Vossen (1994), S. 3.
105 Vgl. Martin (1993), S. 23.
106 Vgl. zum Datenbankentwicklungsprozeß Vossen (1993), S. 48-52.



Quelle: Vossen (1994), S. 48.

Abb. 2.4: Phasen des Datenbank-Entwurfsprozesses

Die Anforderungen sind zuerst zu sammeln und zu dokumentieren und anschließend zu analysieren. Dabei wird zwischen Informationsanforderungen, die festlegen über was Daten gespeichert werden sollen, und Bearbeitungsanforderungen, die festlegen, wie die Daten bearbeitet werden sollen, unterschieden. Diese Phase der *Anforderungsanalyse* entspricht somit der Vorstudie im Life-Cycle-Modell. Die zu erstellende Datenbank muß den Anforderungen an die Datenintegrität, die in Kapitel an ein Schnittstellensystem gestellt werden, genügen..

Ausgangspunkt für die Phase des *konzeptionellen Entwurfs* sind die Anforderungsspezifikationen, aus denen ein abstraktes, zielsystem-unabhängiges

Datenbankschema erzeugt wird. Dazu wird heute zumeist das Entity-Relationship-Modell verwendet. Der Entwurf des konzeptionellen Modells entspricht damit der Anforderungsanalyse im Life-Cycle-Modell und dem Fachkonzept der Datensicht in ARIS.

Nach dem konzeptionellen Entwurf muß eine Entscheidung über das konkret zu verwendende Datenbanksystem (DBS) getroffen werden. Dabei ist neben der vorhandenen Hard- und Software auch die Herstellerbindung des betreffenden Unternehmens ein relevantes Entscheidungskriterium, denn eventuell sind Kompetenzen oder Datenbankmanagementsysteme (DBMS) bereits im Unternehmen vorhanden. Das bei der ABB Kraftwerke AG vorhandene Datenbanksystem ist das relationale System Oracle der Firma Oracle in der Version 7.

In der Phase des *logischen Entwurfs* wird das unabhängige Datenbankschema abhängig vom gewählten Zielsystem z.B. in ein logisches Netzwerk oder ein relationales Datenbankschema überführt. Hierzu bedient man sich Transformationsregeln, um die Konstrukte des konzeptionellen Schemas, z.B. des ER-Modells, in die des Ziel-Datenmodells, im konkreten Fall das relationale Modell, zu übersetzen. Beim *Implementierungsentwurf* muß insb. bei relationalen Modellen auf die Normalisierung[107] geachtet werden. Normalisierung stellt die Redundanzfreiheit von Daten sicher. Durch die Verwendung von ER-Modellen beim konzeptionellen Entwurf ist die Normalisierung bereits sichergestellt.[108]

In der Phase des *physischen Entwurfs* werden unter Berücksichtigung der gestellten Anforderungen bezüglich der Zugriffsarten geeignete Speicherungsstrukturen und Zugriffsmechanismen für die Elemente des logischen Schemas festgelegt. Danach kann die Datenbank mit Hilfe der Datendefinitionssprache (DDL) des konkret verwendeten DBMS (Datanbank Management System) implementiert (eingerichtet) werden.

Prototyping

Beim konventionellen Vorgehen zur Systementwicklung erschweren ungenaue und sich schnell ändernde Anforderungen der Endbenutzer die Systementwicklung. Die im Laufe des Entwicklungsprozesses entstehenden Dokumente sind zu stark auf die Bedürfnisse der Entwickler zugeschnitten um eine geeignete Grundlage für die Kommunikation mit dem Endbenutzer zu bilden.[109] Diese Kommunikationsgrundlage kann durch die Entwicklung von Prototypen geschaffen werden.

---

107 Vgl. Vossen (1993), S. 251ff.
108 Vgl. Barker (1992), S. 148.
109 Vgl. Schönthaler, Nemeth (1989), S. 297.

Ein *Prototyp* ist eine vorläufige, ausführbare Version eines Systems, die die relevanten Merkmale des späteren Produktes aufweist[110] um einerseits dem Endbenutzer zu zeigen, daß die Entwickler die Anforderungen verstanden haben und den Entwicklern zu zeigen, daß die Anforderungen der Benutzer korrekt umgesetzt werden.[111] Er soll den Entwicklern zur Aneignung des Fachwissens dienen, und die Anwender mit den Möglichkeiten des späteren endgültigen Produktes vertraut machen.[112] Unter *Prototyping* wird die Entwicklung, Überprüfung und Bewertung von Prototypen verstanden.[113] Für das Prototyping lassen sich die folgenden drei Ansätze aufführen, die sich durch die verfolgten Ziele unterscheiden:

- exploratives Prototyping
- Ziel dieses Ansatzes ist die Klärung fachlicher Anforderungen durch die Überprüfung auf Konsistenz und Korrektheit und das Aufzeigen verschiedern Lösungsalternativen. Dies geschieht in der Phase der Anforderungsanalyse im Life-Cycle-Modell mit Hilfe von Wegwerfprototypen, deren Entwicklung nicht notwendigerweise in der späteren Zielumgebung erfolgen muß.[114]
- experimentelles Prototyping
- Ziel dieses Ansatzes ist der Nachweis der Tauglichkeit vorgeschlagener Lösungen in jeder Phase des Life-Cycle-Modells bevor aufwendige Implementierungsarbeit in der nächsten Phase geleistet wird. Auch hier werden Wegwerfprototypen eingesetzt, deren Entwicklung nicht notwendigerweise in der späteren Zielumgebung stattfinden muß.[115]
- evolutionäres Prototyping
- Primäre Zielsetzung des evolutionären Prototyping-Ansatzes ist die laufende Anpassung des Systems an geänderte Anforderungen, die in den früheren Phasen noch nicht vorhersehbar sind. Die Prototypen bilden dann eine Folge von

110 Vgl. Schönthaler, Nemeth (1989), S. 298.
111 Vgl. Berztiss (1996), S. 87.
112 Vgl. Schönthaler, Nemeth (1989), S. 299.
113 Vgl. Schönthaler, Nemeth (1989), S.298.
114 Vgl. Schönthaler, Nemeth (1989), S. 304.
115 Vgl. Schönthaler, Nemeth (1989), S. 305f.

Versionen, wobei jede Version als Prototyp seines Nachfolgers dient. Die Prototyp-Entwicklungsumgebung muß vollständig in die Zielumgebung des Anwendungssystems integriert sein. Für das evolutionäre Prototyping bieten sich zwei Verfahren an:

- *inkrementelle Systementwicklung*, bei der das System in den Phasen Implementierung und Systemtest inkrementell erweitert wird, und so jede unfertige Version einen Prototypen darstellt[116] und die
- *evolutionäre Systementwicklung*, bei der der gesamte Prototypingprozeß aus einer Folge der Phasen Anforderungsanalyse, Entwurf, Implementierung und Evaluierung besteht, wobei der Entwickler jederzeit auf neue fachliche Anforderungen reagieren muß und sich der Endbenutzer immer wieder auf neue Versionen einstellen muß.[117]

Für die Unterstützung des Prototyping eignen sich insbesondere datenbankorientierte Entwicklungssysteme, mit denen in kurzer Zeit Datenbankschemata und -abfragen erstellt werden können. Sie unterstützen alle drei genannten Prototyping-Ansätze. Auch operationale Sprachen bei der Spezifikationsformulierung, z.B. Petri-Netze, oder sog. very high-level languages, z.B. Prolog, sind für das Prototyping geeignet, sofern Transformationssysteme für die Überführung in das endgültige System vorhanden sind. Ihr Schwerpunkt liegt jedoch hauptsächlich auf dem explorativen Prototyping.[118]

Modellierungsmethoden

Ein Modell dient als „abstraktes Abbild eines realen Systems für die Zwecke eines Subjektes“[119], i.a. zur Beschreibung eines Problems. Um die Qualität der Modelle über die Erfüllung syntaktischer Regeln hinaus zu erhöhen, muß die Modellierung spezifischen Gestaltungsempfehlungen folgen, den Grundsätzen ordnungsmäßiger Modellierung (GoM). Nach Becker, Schütte (1996) sind dieses der[120]

- Grundsatz der Richtigkeit

116 Vgl. Schönthaler, Nemeth (1989), S. 307.
117 Vgl. Schönthaler, Nemeth (1989), S. 307f.
118 Vgl. Schönthaler, Nemeth (1989), S. 308ff.
119 Becker, Schütte (1996), S. 19.
120 Vgl. zu den GoM Becker, Schütte (1996), S. 65-70.

Das Modell muß sowohl syntaktisch korrekt sein, d.h. mit dem zugrundeliegenden Metamodell konsistent sein, und das abzubildende System korrekt wiedergeben.

- Grundsatz der Relevanz

Jedes Modellelement muß für das Modell von Nutzen sein.

- Grundsatz der Wirtschaftlichkeit

Die Modellerstellung wird wirtschaftlichen Kriterien unterworfen.

- Grundsatz der Klarheit

Hierunter versteht man subjektive Aspekte wie Strukturiertheit, Übersichtlichkeit und Lesbarkeit des Modells, die vom jeweiligen Adressaten abhängig sind.

- Grundsatz der Vergleichbarkeit

Vergleichbarkeit heißt zum einen, daß Modelle, die mit verschiedenen Methoden erstellt wurden, untereinander vergleichbar sind, insb. ineinander überführbar sind (z.B. ER- Modelle in SER[121]- oder SAP-SERM[122]-Modelle). Andererseits soll bei Beachtung dieses Grundsatzes auch die Vergleichbarkeit verschiedener Modelle gleicher Methoden, z.B. in einem Soll-Ist-Vergleich sichergestellt werden.

- Grundsatz des systematischen Aufbaus

Das erstellte Modell muß sich in eine Architektur einfügen, die verschiedene Sichten in einen konsistenten Rahmen fügt.

Datenmodellierung mit Entity-Relationship-Modellen (ERM)

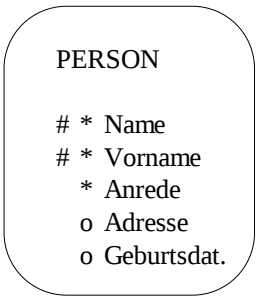
Unterscheidbare Dinge der realen Welt werden Entities genannt. Einzelne Entities, die einander ähnlich sind, werden in einem Entity-Set zusammengefaßt. Entity-Sets werden in Diagrammen durch Rechtecke mit abgerundeten Ecken dargestellt, in denen der Name des Sets angegeben ist.[123] Entities besitzen Eigenschaften, die in der Zusammenfassung zu Entity-Sets Attribute genannt werden. Sie können danach unterschieden werden, ob eine Angabe des Attributwertes notwendig ist, oder ob dieser unbestimmt sein darf. Im ER-Diagramm werden Attribute unter dem Namen des jeweiligen Entity-Sets angegeben. Optionalen Attributen wird ein „o“ vorangestellt.

121 Structured entity relationship.

122 Das SER-Modell der SAG AG.

123 Zu einer Einführung in die verwendete „Krähenfuß-Notation“ vgl. Barker (1992). Eine formalere Einführung in ER-Modelle gibt Vossen (1994), S. 55ff.

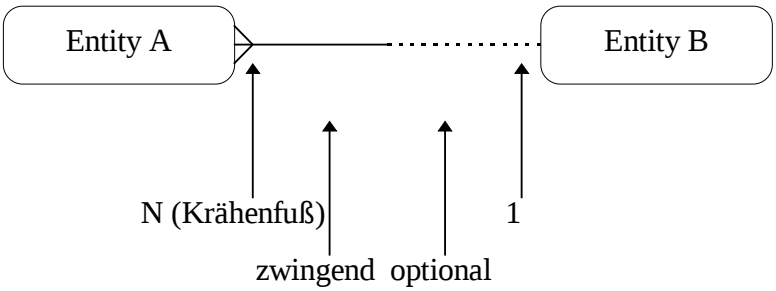
Attribute, deren Wert angegeben werden muß, werden mit einem „*“ gekennzeichnet. Attribute, durch deren Angabe ein Entity eines Entity-Sets eindeutig identifiziert wird, nennt man Schlüsselattribute. In der Darstellung im ER-Diagramm werden Schlüsselattribute durch eine vorangestellte Raute („#“) gekennzeichnet.



Quelle: Barker (1992), S. 43.

Abb. 2.5: Beispiel einer Entity-Set-Darstellung im ER-Diagramm

Verschiedene Entity-Sets stehen in Beziehungen (engl.: relationship) zueinander. Diese werden Relationship-Sets genannt und haben zwei benannte „Enden“. Die Angabe, mit wievielen Entities des ersten Sets ein Entity eines zweiten Sets mindestens oder höchstens in Beziehung stehen kann wird als Komplexität der Beziehung bezeichnet. Im Diagramm werden Relationship-Sets durch eine Linie zwischen zwei Entity-Set-Rechtecke dargestellt. Dabei wird die Komplexität der Beziehung durch die Art der Linie dargestellt (sog. Krähenfuß-Notation):

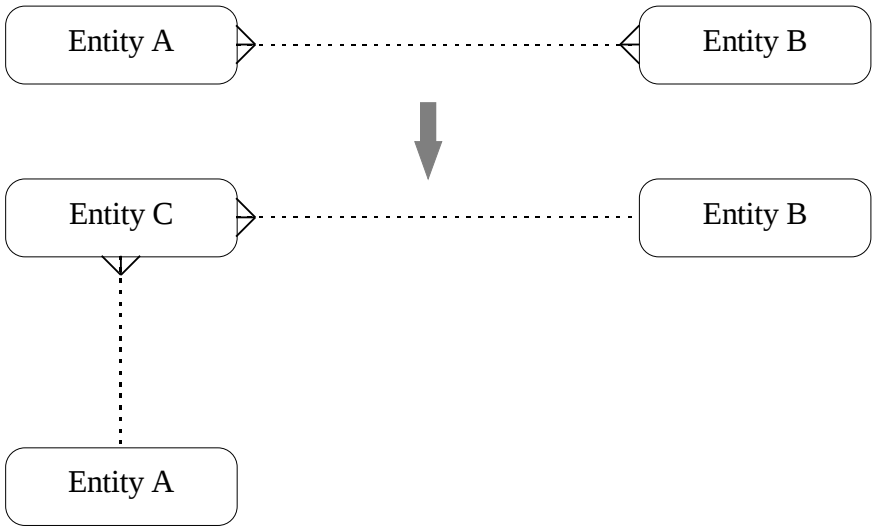


Quelle: Barker (1992), S. 31.

Abb. 2.6: Beziehungstypen in der Krähenfuß-Notation

Wird eine Beziehung über Schlüsselattribute eines Entity-Sets hergestellt, um eine existentielle Abhängigkeit auszudrücken, wird ein Strich durch dasjenige Ende der Beziehung gezogen, dessen Schlüsselattribute in die Beziehung eingehen. N:N-Beziehungen sollten, wenn sie nicht eine einfache gegenseitige Informationsliste

darstellen, durch das Einfügen eines Verbindungsentity-Sets aufgelöst werden. Dadurch wird auch die Zuordnung von Attributen zu Beziehungen ermöglicht.[124]



Quelle: Barker (1992), S. 88.

Abb. 2.7: Auflösung einer N:N-Beziehung

Für Entity-Relationship-Modelle werden nach BECKER, SCHÜTTE (1996)[125] folgende Ausprägungen der GoM angeführt:

GoM	Ausprägungen der GoM für Entity-Relationship-Modelle
Richtigkeit	• Auflistung und Definition der verwendeten Konstrukte • Namenskonventionen • Angabe der Komplexität bei Beziehungen • Explizierung der Spezialisierung nach Vollständigkeit und Disjunktheit • Begriffsbausteine
Wirtschaftlichkeit	• Referenzmodelle • Strukturbausteine • Tooleinsatz

[124] Vgl. Barker (1992), S. 87f.
[125] Vgl. Becker, Schütte (1996), S. 75-82.

Klarheit	• Anordnung der Informationsobjekte von links nach rechts entsprechend des Existenzabhängigkeitsgrads • Minimierung der Anzahl an Kreuzungen der Kanten • Clusterbildung von Datenmodellen • Begriffsbausteine • Strukturbausteine
Vergleichbarkeit	• Begriffsbausteine • Strukturbausteine
Systematischer Aufbau	• Angabe der Komplexität bei Beziehungen • Explizierung der Spezialisierung nach Vollständigkeit und Disjunktheit

Quelle: Becker, Schütte (1996), S. 82.

Tab. 2.1: Exemplarische GoM für Entity-Relationship-Modelle

Prozeßmodellierung mit Petri-Netzen

In dem zu erstellenden Schnittstellensystem müssen verschiedene Prozesse zur Steuerung des Datentransports ablaufen, die als eine zeitliche Abfolge von Funktionen zur Bearbeitung eines Objektes definiert seien[126] und für deren Modellierung sich insb. Petri-Netze eignen.

Der Übergang von der Ablaufspezifikation zur Implementierung ist in der Praxis durch eine hohes Maß manueller Tätigkeiten gekennzeichnet, die selten vollständig dokumentiert und somit schwer nachvollziehbar sind. Operationale Ansätze der Software-Entwicklung umgehen dieses Problem, indem sie für die Anforderungsspezifikation formale Sprachen verwenden, so daß bereits die Spezifikation durch einen Spezifikationsinterpreter ausgeführt werden kann. Dies bietet den Vorteil, daß Spezifikationen sofort, ohne aufwendige Implementierungsarbeit, als Prototyp des Systems dienen können. Änderungen der Software im Wartungsfall werden dadurch wesentlich vereinfacht. Voraussetzung ist die direkte Transformierbarkeit der Spezifikation in das jeweilige Endprodukt. Ein Beispiel einer

126 Vgl. Becker, Vossen (1994), S. 18f.

solchen operationalen Sprache sind Petri-Netze,[127] unter denen man bipartite, direktionale Graphen versteht, die sich durch ein 5-Tupel N=(P, T, I, O, m_0) beschreiben lassen:[128]

$P = \{ p_1, p_2, \ldots, p_n \}, n \neq 0.$
$T = \{ t_1, t_2, \ldots, t_s \}, s \neq 0$ mit $P \cup T \neq \emptyset$ und $P \cap T = \emptyset.$
$I: P \times T \rightarrow \{0,1\}.$
$O: P \times T \rightarrow \{0,1\}.$
$m_0: P \rightarrow \{0,1,2,\ldots\}.$

Die p_i ($1 \leq i \leq n$) sind Plätze, die t_i ($1 \leq i \leq s$) sind Transitionen. I ist eine Input-Funktion die die Menge der gerichteten Kanten von P nach T definiert. O ist eine Output-Funktion, die die Menge der gerichteten Kanten von T nach P definiert und m_0 ist ein Ursprungszustand, wobei die i-te Komponente die Anzahl der Marken auf dem Platz p_i angibt. Für die Zustandsübergänge gelten folgende Regeln:

Eine Transition $t \in T$ heißt aktiv, dann und nur dann wenn $m(p) \neq 0$ wenn
$I(p,t) = 1, \forall p \in P.$
Eine aktive Transition t kann aus dem Zustand m' schalten und erzeugt den neuen
Zustand $m(p) = m'(p) + O(p,t) - I(p,t), \forall p \in P.$

Werden den Plätzen Kapazitäten zugeordnet und die Kanten gewichtet, so spricht man von einem Stellen-Transitionen-Netz. Die Kantengewichte spiegeln dabei wieder, wieviele Marken beim Schalten einer Transition aus den Eingangsplätzen entnommen bzw. in die Ausgangsplätze abgelegt werden.

Diese Stellen-Transitionen-Netze lassen sich erweitern, wenn Marken eine Struktur und unterscheidbare Eigenschaften zugewiesen werden. Die Stellen bezeichnet man dann als Prädikate. Im verwendeten Werkzeug INCOME der Firmat Promatis werden Prädikate auch als Objektspeicher bezeichnet. In dieser Erweiterung als Prädikats-Transitions-Netze genügen sie auch den Anforderungen an die Sichtenintegration, wie sie durch die GoM und die ARIS-Architektur gestellt werden.

Graphisch werden Transitionen als Rechtecke, Stellen oder Prädikate werden als Kreise und Marken als Punkte in den Kreisen dargestellt.

---

127 Vgl. Schönthaler, Nemeth (1990), S. 314f.
128 Vgl. Chao, Zhou, Wang (1994).
Zu einer Einführung in die Petri-Netz-Theorie vgl. Rostenstengel(1989), Reisig (1985).

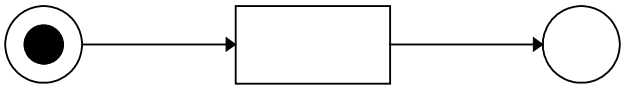


Abb. 2.8: Graphische Darstellung von Petri-Netzen.

Neben unidirektionalen Kanten können in INCOME Transitionen und Objektspeicher auch durch bidirektionale und ungerichtete Kanten verbunden werden. Diese stellen ein Lesen und Schreiben respektive ein Lesen ohne Entnehmen der Marken aus der Stelle dar. Transitionen können durch untergeordnete Petri-Netze hierarchisch verfeinert werden.Verfeinerte Transitionen werden mit zwei senkrechten Strichen gekennzeichnet:

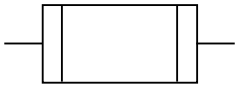


Abb. 2.9: Darstellung verfeinerter Transitionen

Das verfeinernde Netz enthält sämtliche Stellen des Vor- und Nachbereiches der verfeinerten Transition durch die die Verbindung zum übergeordneten Netz hergestellt wird. Diese Stellen sind graphisch durch ein den Kreis umschließendes Rechteck gekennzeichnet.

Um die Übersichtlichkeit des Netzes bei graphischer Darstellung zu wahren, bietet INCOME das Konzept der Rollen an. Derselbe Objektspeicher läßt sich damit an mehreren verschiedenen Orten im Netz plazieren. Die Bezeichnung einer solchen Rolle besteht aus dem Namen des Objektspeichers, ergänzt um eine fortlaufende Numerierung. Graphisch werden die Rollen durch gestrichelte Kreise dargestellt.

In Anlehnung an BECKER, SCHÜTTE (1996) können für Petri-Netze exemplarische GoM angeführt werden[129]:

GoM	Ausprägungen der GoM für Prozeßmodelle
Richtigkeit	• Auflistung und Definition der verwendeten Konstrukte • Namenskonventionen • Konsistenz der Semantik des Prozeßmodells mit dem Daten- und Funktionsmodell
Relevanz	• Verzicht auf Organisations- und Anwendungssystemsymbole

[129] Vgl. Becker, Schütte (1996), S. 82-92.

Wirtschaftlichkeit	• Referenzmodelle • Tooleinsatz
Klarheit	• Anordnung der Informationsobjekte entsprechend des Durchlaufes von oben nach unten. • Minimierung der Anzahl an Kreuzungen der Kanten • Nutzung von Verfeinerungen
Vergleichbarkeit	• Namenskonventionen • Strukturbausteine
Systematischer Aufbau	• Beachtung der Existenzabhängigkeiten des Datenmodells

Quelle: Angelehnt an Becker, Schütte (1996), S. 92.

Tab. 2.2: Exemplarische Ausprägungen der GoM für Prozeßmodelle

In einem Petri-Netz-Modell der Ablaufsteuerung ist die Funktionshierarchie als Teil der Funktionssicht abgebildet. Die Prädikat-Transitions-Netze besitzen viele Elemente und Eigenschaften von Datenflußdiagrammen, die im Rahmen der SA (structured analysis) und verwandter Techniken zur Darstellung der Datenflüsse zwischen den Funktionen und zwischen Funktionen und Datenspeichern verwendet werden[130] und auf deren Modellierung daher verzichtet wird.

Werkzeugbedingte Restriktionen

Durch Zuordnung der Prädikate zu den im Datenmodell definierten Entity-Sets kann eine Integration der Daten- und der Steuerungssicht vorgenommen werden. Die Integration der mit INCOME modellierten Prozesse mit den Daten im verwendeten CASE-Werkzeug geht jedoch nicht soweit, daß aus den Beschreibungen der Funktionen der Prozesse Funktionen und Module im CASE-Werkzeug erzeugt werden können. Zwar können Funktionsdekompositionsdiagramme erzeugt werden, doch die hinterlegten Aussagen der Prädikatenlogik, konkret wird in INCOME die Sprache Prolog verwendet, lassen sich dabei nicht übernehmen. Dieses ist auch im Zusammenhang mit den großen Unterschieden der verwendeten Sprachen zu sehen: INCOME verwendet mit Prolog eine Form der Prädikatenlogik während das CASE-

130 Vgl. Schönthaler, Nemeth (1990), S. 62ff.

Werkzeug deklarativ prozedurale Sprachen wie PL/SQL, um prozedurale Konstrukte ergänzte SQL, unterstützt. Die Programmierung mit Prädikatenlogik in Prolog besteht aus einer Beschreibung des zu lösenden Problems, während in prozeduralen Sprachen einen Weg zur Problemlösung angegeben wird. Diese beiden Programmieransätze sind somit inhärent inkompatibel. Die Modellierung der Prozesse in Petri-Netzen kann daher nur zur Dokumentation des Prozeßablaufes dienen. Eine Eignung als Prototyp auf Spezifikationsebene ist zweifelhaft, da eine direkte Transformation oder eine Weiterentwicklung des Petri-Netz-Modells in ein einsatzfähiges Endprodukt derzeit nicht möglich ist. Dies ist aber ein wichtiger Beitrag operationaler Sprachen zum evolutionären Prototyping[131]. Eine umfangreiche Implementierung der Prozesse sowohl in Prädikatenlogik als auch in prozeduraler Sprache in der Zielumgebung des evolutionären Prototypen ist weder terminlich noch wirtschaftlich vertretbar, da derzeit vom Werkzeug keine einfache Transformationsmöglichkeit angeboten wird.

Als Beschränkung im Verlauf der Implementierung erweist sich der durch das Werkzeug Designer/2000 der Firma Oracle zur Verfügung gestellte Programmgenerator. Dieser ist in zwei für die Realisierung des Schnittstellensystems zentralen Bereichen mangelhaft. Diese Mängel werden verstärkt durch das Fehlen einer ausreichenden Dokumentation dieses Werkzeugs. Zum einen werden Dateibearbeitungen nur unzureichend unterstützt. Ein Großteil der Schnittstellenlogik befaßt sich mit dem Lesen und Schreiben von linearen Übergabedateien. Das verwendete Werkzeug Designer/2000 unterstützt primär die Datenbearbeitung in relationalen Datenbanken, die wichtiger Bestandteil des Schnittstellensystems ist, aber durch zahlreiche Ein- und Ausgabemodule ergänzt werden muß. Bei den betoffenen Modulen, die sich primär mit der Bearbeitung von Dateien beschäftigen, muß daher auf eine Werkzeugunterstützung verzichtet werden. Diese Module werden manuell implementiert. Auch Anforderungen an eine zeitgemäße Benutzeroberfläche des Schnittstellensystems mit den von Seiten der Fachabteilung geforderten Funktionalitäten und Bedienkomforts können durch den zur Verfügung stehenden Codegenerator nicht erfüllt werden. Die verfügbaren Funktionen zur Modulmodellierung und -generierung des Designer/2000 können für dieses Projekt daher nicht genutzt werden.

Nicht zuletzt aufgrund der Unausgereiftheit und der häufig auftretenden Fehler der Codegenerators wird auch auf eine nur teilweise CASE-Werkzeug unterstützte Modellierung und Implementierung der Ablauflogik des Schnittstellensystems verzichtet. Die funktionalen Aspekte sind jedoch zu einem großen Teil den Prozeßmodellen zur Ablaufsteuerung zu entnehmen.

---

131 Vgl. Schönthaler, Nemeth (1993), S. 307.

Umsetzung am Beispiel

Analyse der bestehenden Systeme

Entsprechend dem in dargestellten Vorgehensmodell muß zuerst der Ist-Zustand im Systemumfeld aufgenommen werden. Daher werden die für das zu realisierende System relevanten bestehenden Systeme und deren Schnittstellen aufgeführt. Die Systeme werden anhand der Reihenfolge der Funktionen des Auftragsbearbeitungsprozesses beschrieben, um eine Einordnung in den übergeordneten Planungsablauf (vgl. Anhang) zu ermöglichen.

Übergeordnete Planung

Nachdem ein Kundenauftrag über eine Kraftwerkskomponente bei der KWE eingeht, wird auf der Basis technischer Machbarkeit geprüft, welche Teile hiervon durch die ABB Kraftwerke AG gefertigt werden können und welche von anderen ABB Gesellschaften bezogen werden müssen. Sind die Herstellteile identifiziert, werden Aufträge mit Grobterminen für diese Teile an die Bereiche KW/PG und KW/PT gegeben. Diese Bereiche prüfen die Machbarkeit aufgrund grober kapazitiver Restriktionen und können ihrerseits Fremdbeschaffungsaufträge auslösen. Für die Herstellung der mechanischen Komponenten erhält der Bereich KW/PM einen Auftrag mit Terminen und Konstruktionsdaten. Stücklistendaten werden durch ein gemeinsam benutztes EDV-System bereichsübergreifend zur Verfügung gestellt. Zeichnungen liegen entweder in Papierform oder in elektronischer Form vor.

Das PPS-System

Alle Unterbereiche des Bereichs KW/P planen und steuern mit dem PPS-System Triton der Firma Baan in der Version 3.1. Triton ist eine konfigurierbare und erweiterbare Standardsoftware, die sich individuell auf die Untenehmensbedarfe anpassen läßt und die als hostbasierte Anwendung auf Basis einer relationalen Datenbank implementiert ist. Der Zugriff erfolgt über alphanumerische Terminalemulationen vom PC aus.

Abbildung der Aufbauorganisation in Triton

Die Unterbereiche des Bereiches KW/P sind als Mandanten innerhalb des PPS-Systems abgebildet. Diese sind in Hauptabteilungen und diese wiederum in Unterabteilungen gegliedert. Den Unterabteilungen können Maschinen zugeordnet werden. Eine

Fertigungsinsel im Bereich KW/PM ist als Hauptabteilung abgebildet, die wesentlichen, kapazitiv kritischen Maschinengruppen einer Fertigungsinsel sind als Unterabteilungen abgebildet und einzelne Maschinen, nicht alle, als Maschinen.

Sowohl Abteilungen als auch Maschinen bieten eine Kapazität an, die bei Maschinen in Maschinenstunden pro Zeiteinheit (z.B. pro Woche) gemessen wird. Abteilungen kann sowohl eine Mitarbeiterkapazität (Anzahl der zugeordneten Mitarbeiter) als auch eine maschinelle Kapazität (Maschinenstunden pro Zeiteinheit) zugeordnet sein. Das maschinelle Kapazitätsangebot kann aus den Kapazitätsangeboten der der Abteilung zugeordneten Maschinen aggregiert sein. Verfügen Abteilungen über zugeordnete Maschinen, muß als kritische Kapazitätsart entweder die Mitarbeiter- oder die Maschinenkapazität gewählt werden. Die kritische Kapazität ist diejenige, die im Normalfall zuerst einen Kapazitätsengpaß in der Abteilung erzeugt und aufgrund derer das PPS-System disponieren soll.

Projektdaten in Triton

Aufgrund der Komplexität der Erzeugnisse und der Berücksichtigung von kundenspezifischen Anforderungen erfolgt die Planung innerhalb KW/P projektorientiert. Kundenaufträge werden in PG-, PM- und PT-Komponenten strukturiert und innerhalb Triton in einzelne Projekte überführt. Im PPS-System existiert neben den projektgebundenen Datenbereichen ein projektneutraler Datenbereich für die anonyme Fertigung von Teilen niedriger Dispositionsstufen. Daten können zwischen den projektbezogenen Bereichen und dem neutralen Bereich kopiert werden.

Nachdem ein Projekt für die Herstellung eines Erzeugnisses angelegt ist, werden diesem Projekt Aktivitäten zugeordnet, die wesentliche Bestandteile eines Projektes darstellen und mit einer bestimmten Dauer versehen sind. Aktivitäten sind z.B. Arbeitsplanerstellung, Fertigung von Hauptbestandteilen, Endmontage etc. Aktivitäten sind die Nachfrager der von den Abteilungen angebotenen Kapazitäten. Sie stehen innerhalb eines Projektes untereinander in Vorgänger-Nachfolger-Beziehungen und bilden so einen Projektnetzplan, der zur Terminierung des Projektes eingesetzt werden kann. Durch Methoden der Netzplantechnik können, ausgehend von einem einzuhaltenden Endtermin für das Projekt, den Aktivitäten Start- und Endzeitpunkte und Pufferzeiten zugewiesen werden, innerhalb derer sie auszuführen sind. Bei der Betrachtung auf dieser Abstraktionsebene werden die von den Abteilungen angebotenen Kapazitäten nicht automatisch berücksichtigt. Eine Berechnung von nachgefragten Kapazitäten pro Periode ist möglich und Über- und Unterlasten können in einfacher

graphischer Form angezeigt werden. Die zeitliche Verschiebung von Aktivitäten im Netzplan zum Zweck des Kapazitätsabgleichs ist manuell möglich.

Jeder Aktivität kann eine Baugruppe zugeordnet werden. Unter einer Baugruppe wird ein Teil verstanden, welches für das Projekt von herausragender Bedeutung ist. Bei Turbinen sind dieses z.B. die Gehäuse, Leitschaufeln etc. Baugruppen besitzen eine komplette Stücklistenstruktur bis zu Rohmaterial bzw. Fremdbezugteilen. Durch die Assoziation von Baugruppen mit Projektaktivitäten wird die Voraussetzung für die weitere material- und zeitwirtschaftliche Betrachtung des Projektes geschaffen.

Arbeitspläne in Triton

Für jedes Herstellteil werden in Triton ein oder mehrere Arbeitspläne geführt, die durch eine Nummer und die Angabe des Teils, dessen Herstellung sie beschreiben, identifiziert werden. Jeder Arbeitsplan besteht aus Arbeitsplanpositionen. Jede Arbeitsplanposition bildet sich aus der Zuordnung eines arbeitsplanneutralen Arbeitsganges zu einem Arbeitsplan. Innerhalb jedes Arbeitsplanes sind die Arbeitsplanpositionen eindeutig durch einen Index gekennzeichnet, der auch die Reihenfolge der Bearbeitung angibt. Die Arbeitsplanpositionen sind mit zeitlichen Gültigkeitsdaten versehen, sodaß zeitabhängige Arbeitspläne gebildet werden können. Jeder Arbeitsgang ist sowohl einer Abteilung als auch, bei maschinellen Arbeitsgängen, einer Maschine zugeordnet. Diese Zuordnung wird i.a. in die Arbeitsplanposition übernommen, kann aber arbeitsplanspezifisch geändert werden. Ebenso können die den Arbeitsgängen zugeordneten Kapazitätsbedarfe in die Arbeitsplanpositionen übernommen oder aber geändert werden. Das PPS-System kennt Alternativarbeitspläne nur in der Form, als daß sich zu einem Teil losgrößenabhängige Arbeitspläne zuordnen lassen. Funktionen zum Verwalten von Arbeitsfolgen als Teile von Arbeitsplänen sind nicht vorhanden.

Bearbeitungsplätze, Werkzeuge und Vorrichtungen in Triton

Im Triton-System können weder Werkzeuge noch Vorrichtungen verwaltet werden und sind damit auch nicht kapazitiv behandelbar. Das System unterstützt Fertigungshilfsmittel nur in einer informationellen Form. Da sie nicht kapazitiv disponibel sind, muß sichergestellt sein, daß sie keine Engpaßressourcen darstellen. Fertigungshilfsmittel sind durch einen Fertigungshilfsmitteltyp gekennzeichnet. Eine Beistellung der Fertigungshilfsmittel kann nur für Arbeitsplanpositionen angegeben werden, nicht für die arbeitsplanneutralen Arbeitsgänge.

Eine andere Möglichkeit einer Arbeitsplanposition Fertigungshilfsmittel beizustellen ist die Angabe dieser Mittel in der Stückliste. Hier ist die Führung von Stücklistenpositionen als sog. Zusatzpositionen möglich. Als Zusatzpositionen können fiktive Artikel geführt werden, für die bei einer Bedarfsermittlung keinerlei Fertigungs-, Lager- oder Bestellvorschläge generiert werden.

In der Praxis stellen die Bearbeitungsplätze von Maschinen Engpässe dar, u.a. aufgrund der Möglichkeit des hauptzeitparallelen Rüstens auf diesen Plätzen und müssen kapazitiv behandelt werden. Da das PPS-System aber eine kapazitive Behandlung nur für Maschinen zuläßt, werden die zugehörigen Bearbeitungsplätze als Fertigungshilfsmittel beigestellt. Diese Einordnung kann durch den Charakter von Bearbeitungsplätzen gerechtfertigt werden, denn die eigentliche Bearbeitung wird von der Maschine übernommen. Eine kapazitive Behandlung weiterer kritischer Ressourcen, neben Maschinen, muß daher auf einer untergeordneten Planungsebene erfolgen. Dort müssen sowohl Werkzeuge, als auch Vorrichtungen und Bearbeitungsplätze, soweit notwendig, in die Kapazitätsplanung eingehen.

Auftragsterminierung in Triton

In einem Projekt können im Rahmen der Material- und Zeitwirtschaft aus den Aktivitäten Fertigungsaufträge gewonnen werden. Über die mit den Aktivitäten verknüpften Baugruppen läßt sich über eine Stücklistenauflösung der Sekundärbedarf für verschiedene Planungsperioden ermitteln. Ausgehend von Wiederbeschaffungszeiten bei Fremdbezugteilen oder den Arbeitsgangzeiten bei Herstellteilen lassen sich Mengen und Termine für Bestellvorschläge bzw. Fertigungsauftragsvorschläge ermitteln. Es ist möglich, Herstellteilen, denen kein Arbeitsplan zugeordnet ist, ebenfalls eine Wiederbeschaffungszeit zuzuordnen, mittels derer die Fertigungsauftragsvorschläge terminierbar sind. Dies ist z.B. bei noch nicht vollständigen Projektunterlagen notwendig, um möglichst früh mit der Produktionsplanung beginnen zu können.

Die Bildung der Start- und Endtermine für Fertigungsauftragsvorschläge und -auftragspositionen erfolgt als Rückwärtsterminierung durch einen sog. PRP-Lauf (project requirements planning), zunächst ohne Berücksichtigung der verfügbaren Kapazitäten. Ein PRP-Lauf kann entweder als kompletter Neuaufwurf oder als net-change- Planung durchgeführt werden. Er kann sowohl manuell durch den Bediener ausgelöst werden, als auch periodisch in frei definierbaren Zeitintervallen durch das PPS-System automatisch gestartet werden.

Das Ergebnis des PRP-Laufes sind Lagerauftrags-, Bestell-, und Fertigungsauftragsvorschläge. Mit diesen Vorschlägen können Abteilungs- und Maschinenauslastungen und geplante Lagerbewegungen auf Realisierbarkeit analysiert werden. Die Fertigungsauftragsvorschläge sind als GANTT-Diagramm darstellbar. Eine Darstellung als Kapazitätsgebirge ist möglich, wobei eine graphische Gegenüberstellung mit den Kapazitätsangeboten der Maschinen erfolgt. Der Bediener kann mit der grafischen Darstellung Engpässe identifizieren und korrigierend eingreifen, indem die Vorschläge z.B. durch zeitliches Verschieben, Splitten, Raffen etc, geändert werden. Die Auftragsvorschläge werden durch Bestätigung und explizite Überführung in Aufträge umgewandelt.

Bei der Erzeugung von Fertigungsauftragsvorschlägen werden alle Arbeitsplanpositionen auf den Fertigungsauftragsvorschlag kopiert und dort mit Mengenangaben aus der Stücklistenauflösung versehen. Eine Auftragsposition gibt daher neben der Art der Bearbeitung auch die zu fertigende Menge an und kann unabhängig vom zugrundeliegenden Arbeitsplan geändert werden. Alle Auftragspositionen eines Fertigungsauftrages sind mit Start- und Endterminen versehen und können für die Fertigung freigegeben werden.

Bei der Fertigungsfreigabe nimmt Triton keine Materialverfügbarkeitsprüfung vor. Nachdem Fertigungsaufträge freigegeben sind, können daran keine Änderungen bezüglich der Reihenfolge der Auftragspositionen vorgenommen werden. Eine solche Änderung muß über die Stornierung des Auftrags geschehen. Danach muß manuell ein neuer Auftrag aus einem neuem Arbeitsplan generiert werden, der dort ansetzt, wo der ursprüngliche Auftrag abgebrochen wurde.

Feinplanung in Triton

Feinplanungsfunktionen für eine Maschinenbelegungsplanung sind in Triton nur unzureichend vorhanden. Zum einen ist in Triton keine kapazitive Behandlung von Fertigungshilfsmitteln möglich. Dies führt dazu, daß zwar bei den Maschinen Engpässe erkannt werden, bei den Bearbeitungsplätzen, sofern sie nicht in einer 1:1-Beziehung zu einer Maschine stehen, keine Engpaßidentifizierung möglich ist. Gleiches gilt für Fertigungshilfsmittel. Hier sind die Kapazitätsrestriktionen in der Praxis jedoch selten kritisch, da Werkzeuge mehrfach vorhanden oder bei Bedarf aus Komponenten mehrfach modular zusammenstellbar sind.

Die kleinste zeitliche Einheit des Triton-Systems ist ein Tag. Eine tagesgenaue Planung ist auf der Ebene von Fertigungsaufträgen mit Bearbeitungs- oder Durchlaufzeiten im

Wochenbereich ausreichend. Für Aufträge mit wesentlich geringeren Bearbeitungszeiten, z.B. im Schicht- oder Stundenbereich, ist eine tagesgenaue Planung für eine Maschinenbelegung allerdings nicht ausreichend. Müssen die aus dem PRP-Lauf resultierenden Planungen verfeinert werden, so muß dieses in untergeordneten Systemen erfolgen.

Da das Triton-System in der Lage ist, die Grobplanung von der Projektebene bis auf die Fertigungsauftragsebene durchzuführen und die Verwaltung von Grunddaten wie Stücklisten und Arbeitspläne zu übernehmen, ist ein nachrangiges Feinplanungssystem in der Lage, die mit Terminen übergebenen Fertigungsaufträge und zugeordneten Arbeitspläne zu übernehmen und eine fertigungsinselinterne Maschinenbelegungsplanung unter kapazitiver Berücksichtigung aller potentiell knappen Ressourcen durchzuführen.

Die Leitstände

Die Fertigungsinseln im Bereich KW/PM setzen zur Feinplanung die Fertigungsleitstände (FLS) Soflex der Firma Soflex ein. Jede der Fertigungsinseln verfügt über ein unabhängiges FLS-System. Soflex ist als eine grafikfähige X/Windows-Anwendung unter dem Betriebssystem VMS der Firma Digital Equipment Corp. realisiert. Der Zugriff erfolgt über Standard PC mittels X/Windows Emulation.

Soflex steht mit weiteren Systemen in Verbindung, insb. dem Werkzeugverwaltungssystem TDM (Tool Data Management) der Firma Walter Informationssysteme, über das Werkzeugbedarfs- und Werkzeugbereitstellungsmeldungen erfolgen. Soflex beherrscht die direkte Kommunikation mit den Steuerungen der Maschinen und kann als DNC-Rechner dienen.

Datenverwaltung in Soflex

Soflex besitzt keine Stammdatenverwaltung für Stücklisten, Arbeitspläne und Fertigungsaufträge. Es werden auch keine Werkzeugdaten oder NC-Programme dauerhaft geführt. Soflex speichert nur Maschinen- und Bearbeitungsplatzdaten auftragsunabhängig. Das System verfügt über eine Zuordnung der auf jedem Platz einsetzbaren Maschine und kann so beide Ressourcen kapazitiv behandeln. Alle weiteren Daten müssen dem System in Verbindung mit Fertigungsaufträgen zur Verfügung gestellt werden.

Die an Soflex zu übergebenden Daten sind die Fertigungsaufträge selbst, die dazugehörigen Auftragspositionen[132], die den Auftragspositionen zugeordneten NC-Programme, die Liste der möglichen Bearbeitungsplätze für jede Auftragsposition, eine Werkzeugeinsatzliste für jedes NC-Programm und die Fertigungshilfsmittelliste für jede Auftragsposition. Bearbeitungsplätze werden als Fertigungshilfsmittel betrachtet.[133] Diese auftragsbezogenen Daten werden spätestens 24 Stunden nach Beendigung des Auftrages automatisch gelöscht.

Feinplanung in Soflex

Soflex erzeugt einen von der geplanten Verfügbarkeit der Fertigungsressourcen abhängigen Maschinenbelegungsplan. Es wird für jede Maschine eine Folge von Auftragspositionen auf der Basis der freigegebenen Fertigungsaufträge und Auftragspositionen generiert. Soflex kennzeichnet Fertigungsaufträge mit zwei Freigaben, einer Planungsfreigabe und einer Fertigungsfreigabe. Diese spielen eine Rolle bei der Belegungsplanung, die in Soflex in zwei Schritten erfolgt:

Im ersten Schritt wird ein Simulationsplan erstellt, in den alle für die Planung freigegebenen Fertigungsaufträge eingehen. Dazu zählen auch bereits eingeplante Fertigungsaufträge, bis zu einem vom Benutzer wählbaren Termin. So wird sichergestellt, daß bereits eingeplante Auftragspositionen nicht kurzfristig umgeplant werden, andererseits bleibt die Möglichkeit eines kompletten Neuaufwurfs der Planung erhalten. Ein vorhandener Auftragsvorrat wird auf einzelne Bearbeitungsplätze und Maschinen[134] eingeplant. Diese Planung erfolgt nach den folgenden Prioritätsregeln auf Ebene der Auftragspositionen. Ausgehend vom geforderten spätest zulässigen Fertigstellungstermin des Auftrages ermittelt Soflex durch eine Rückwärtsrechnung unter Verwendung der Mengen und der Bearbeitungszeiten der Auftragspositionen das spätest mögliche Datum für den Bearbeitungsbeginn. Dieses Datum ist das vorrangige Einplanungskriterium; höchste Priorität hat der Fertigungsauftrag mit dem frühesten spätest möglichem Anfangstermin. Bei gleichen Terminen, die Betrachtung erfolgt tagesgenau, wird die Reihenfolge durch Angabe einer expliziten Prioritätskennzahl der Aufträge festgelegt. In der Fertigungsinsel „Großmechanik“ wird als weiteres Kriterium die Werkzeugbestückung der Maschinen herangezogen. Der Arbeitsgang, der die wenigsten Werkzeugein- und
-auslagerungsoperationen benötigt, hat die höhere Priorität.

---

132 In der Soflex Terminologie wird zwischen „Aufträgen“, die nur den Auftragskopf darstellen, und dem „Arbeitsplan“, der die Auftragspositionen enthält, unterschieden.

133 Vgl. Kapitel , Das PPS-System.

134 In der Soflex Terminologie heiß die Maschine Station.

Im zweiten Schritt kann ein Simulationsplan in einen aktiven Maschinenbelegungsplan überführt werden, oder er kann in einen bereits vorhandenen übernommen werden. Bei der Überführung oder Übernahme können nur die Aufträge übernommen werden, für die die Fertigungsfreigabe vorliegt.

NC-Programme in Soflex

Nach der Belegungsplanung ermittelt Soflex aus den Informationen in den Auftragspositionen und den aus dem Belegungsplan entnommenen Terminen den Bedarf an NC-Programmen und fertigungsunterstützenden Daten (FUD) im Zeitablauf. Zu jeder Auftragsposition kann maximal ein NC-Programm angegeben werden, es sind beliebig viele FUD möglich. Der NC-Programmbedarf wird dem NC-Programmiersystem gemeldet. Unter einem NC-Programm werden neben Kopfinformationen (z.B. Ersteller, Erstelldatum) und dem DIN-Code nach DIN 66025 auch die Bedarfslisten für Werkzeuge (Werkzeugeinsatzliste), Vorrichtungen (Vorrichtungseinsatzliste) und Aggregate (Aggregateeinsatzliste) verstanden. Ein Aggregat stellt eine Vorrichtung zum Fixieren des Werkzeugs in der Maschine dar. Daneben enthält ein NC-PRogramm die Angabe der Maschinen, auf denen der Code ablauffähig ist und einen beschreibenden Text.

Nach der Bereitstellung des NC Programmes ermittelt Soflex aus der Werkzeugeinsatzliste und den geplanten Startzeiten der Auftragspositionen den Werkzeugbruttobedarf im Zeitablauf. Dieser wird mit der Planbestückung der Werkzeugmagazine der Maschinen abgeglichen und ein Werkzeugnettobedarf errechnet. Der Werkzeugbestand wird bei Ein- oder Auslagerung von Werkzeugen in Soflex aktualisiert.

Werkzeugbereitstellung

Die Werkzeugbereitstellung erfolgt mit Unterstützung des Werkzeugdatenverwaltungssystems TDM (Tool Data Management) der Firma Walter Informationssysteme. Soflex meldet den Werkzeugnettobedarf an TDM. Die benötigten Werkzeuge werden aus Komponenten montiert, eingestellt und vermessen und in die Werkzeugmagazine der Maschinen eingelagert. Das Einstellgerät meldet die Ist-Daten der Werkzeuge, insb. die Reststandzeit, an TDM. Dieses gibt die Daten an Soflex weiter und meldet damit die Bereitstellung der Werkzeuge.

Auslösung der Bearbeitung

Zum Auslösen der Bearbeitung ist das benötigte Material bereitzustellen. Soflex prüft die Materialverfügbarkeit aufgrund der Rückmeldungen der vorhergehenden Arbeitsgänge, bzw. des Materialeinschleusvorganges, mit dem Material physisch und informationstechnisch in die Fertigungsinsel eingeführt wird. Wenn das benötigte Material verfügbar ist, überträgt Soflex das entspechende NC-Programm in die Maschinensteuerung. Je nach Steuerungstyp und Freigabestatus des NC-Programmes muß der Start des Programms evtl. manuell ausgelöst werden.

NC-Programm Rückübertragung

Der Maschinenbediener hat die Möglichkeit, während der Bearbeitung den DIN-Code des NC-Programmes in der Maschinensteuerung zu ändern, z.B. wegen unvorhersehbarer Schwankungen der Materialbeschaffenheit oder zur Verfahrwegoptimierung. So kann das Fachwissen des Maschinenbedieners in die Optimierung der NC-Programme eingehen. Der geänderte DIN-Code wird nach Ende der Bearbeitung aus der Maschinensteuerung an Soflex rückübertragen. Eine der beiden in Soflex vorhandenen Versionen kann als weiterhin gültige Version übernommen werden, die andere wird automatisch gelöscht. Soflex stellt geänderte NC-Programme dem NC-Programmiersystem zur Verfügung, damit der Programmierer Veränderungen im DIN-Code im Quellcode manuell nachtragen kann.

NC-Programmiersystem

NC-Programme werden mit einem NC-Programmiersystem[135] steuerungstypenunabhängig erzeugt. Sie sind maschinenabhängig, insb. in bezug auf die Abmessungen der Bearbeitungsplätze. Die Menge der möglichen Maschinen wird durch die Geometrie der Werkstücke und durch technologische Faktoren eingeschränkt. Obwohl zu einem großen Teil Bearbeitungszentren und flexible Fertigungssysteme eingesetzt werden, unterscheiden sich einige Maschinen oder Bearbeitungsplätze durch die Art der auf ihnen zulässigen Bearbeitungsart.[136]

Der Quellcode des NC-Programms wird durch einen Postprozessor unter Angabe der Maschine und Steuerung in ein Programm nach DIN 66025 übersetzt. Der Postprozessor erzeugt dabei auch die Listen der im Laufe der Programmabarbeitung

[135] z.Z. ist Exapt bzw. Exapt+ im Einsatz.

[136] Dies ist z.B. bei den Doppel-Gantry Anlagen in der Großmechanik der Fall: Auf den Karussellplätzen können Dreh-, Bohr- und Fräsbearbeitungen stattfinden, auf den Plattenfeldplätzen sind nur Bohr- und Fräsarbeitsgänge möglich.

benötigten Werkzeuge, Vorrichtungen und Aggregate in einem Format, wie es vom FLS-System verarbeitet werden kann.

Schnittstellen der bestehenden Systeme

Eine DV-technische Integration des übergeordneten PPS-Systems mit den Fertigungsleitständen besteht nicht. Die Fertigungsleitstände werden zur Zeit nur für den DNC-Betrieb genutzt. Die über eine Netzwerkverbindung an sie übertragenen NC-Programme werden auf Anweisung des Maschinenbedieners in die jeweilige Steuerung übertragen. Die Fertigungsleitstände sind zusammen mit dem TDM-System in der Lage, den Werkzeugkreislauf zu steuern. Da keine Auftragsdaten übermittelt werden, können die Leitstandsysteme weder eine Maschinenbelegungsplanung noch eine Werkzeugbedarfsrechnung im Zeitablauf durchführen. Derzeit werden Aufträge wöchentlich fertigungsinsel-übergreifend gesammelt und von den Fertigungsinselleitern werden nach Absprache Ecktermine für die Aufträge festgelegt, die dann in der Insel durchzusetzen sind.

Schnittstellen des PPS-Systems

Das PPS-System verwaltet seine Daten in einem relationalen Datenbanksystem. Ein direkter Zugriff aus externen Programmen auf die Daten ist somit prinzipiell möglich. Weil das zu erstellende Schnittstellensystem aber nicht spezifisch für das zur Zeit eingesetzte PPS-System erstellt werden soll, muß auch hier eine Dateischnittstelle vorhanden sein, die die benötigten Daten aus den Tabellen liest und in dem folgenden Format zur Verfügung stellt:

Dateinamen

Die Dateinamen müssen folgende Endungen besitzen:

Dateiinhalt	Dateinamenendung
Arbeitsplan	APL
Fertigungsauftrag	AUP

Tab. 3.1: Dateinamen der Triton-Schnittstelle

Die zu übergebenden Daten stehen in den textuellen Übergabedateien jeweils in einer eigenen Zeile. Die Daten werden durch Textkürzel am Anfang der Zeile identifiziert.

Die Textkürzel sind 15 Zeichen lang, ggf. wird das Kürzel mit dem Zeichen „.“aufgefüllt. Nach dem Textkürzel folgt ein Doppelpunkt, nach diesem der Wert. Die Reihenfolge der Zeilen ist beliebig. Da die Dateien sequentiell gelesen werden, wird bei mehrfach angegebenen Daten jeweils das zuletzt gelesene gespeichert.

Auftragsdateien

Da ein Fertigungsauftrag immer mit einem Arbeitsplan zusammen übertragen werden muß, muß dies in derselben Datei erfolgen, in der der Arbeitsplan nach dem Fertigungsauftragskopf folgt. Die Fertigungsauftragsnummer muß in der ersten Zeile der Datei angegeben sein, die Reihenfolge der anderen Daten ist beliebig.

Triton Begriff	Pflicht	Format	Textkürzel	Bemerkung
Produktions-auftrag	J	CHAR(13)	FTGAUFTRAG	Identifizierendes Merkmal
Planer	J	CHAR(15)	ERSTELLER	Ersteller des Auftrags (Name)
Eingangsdatum	J	JJJJ.MM.DD: HH.MM.SS	ERSTELLDZ	
Bezeichnung	N	CHAR(60)	KOMMENTAR	
Bestellte Menge	J	INTEGER(10)	STUECK	Anzahl der zu fertigenden Werkstücke
Bezeichnung Artikelcode	J	CHAR(60)	BEZEICHNG	
Projekt	J	CHAR(9)	PROJEKT	
Lieferdatum	J	JJJJ.MM.DD: HH.MM.SS	LIEFERUNG	Anlieferungstermin des Materials
Artikelcode	J	CHAR(16)	HERST_ID	
Mandant	J	INTEGER(3)	MANDANT	Der Mandant der den Auftrag auslöst
Fertigungsstufe	N	CHAR(3)	FTGSTUFE	Die Fertigungsstufe der

(Aktivitäten entsprechen den Fertigungsstufen)				der Auftrag zugeordnet ist
Lager	N	CHAR(8)	ABLIEFERORT	Der Ablieferort, an dem das Material bei Beendigung des Auftrags zur Verfügung zu stellen ist. (Bei inselübergreifenden Aufträgen wird hier der Name der nächsten Fertigungsinsel eingetragen)

Tab. 3.2: Auftragsdaten der Triton-Schnittstelle

Arbeitspläne

Arbeitspläne werden zusammen mit den Arbeitsplanpositionen aus einer Datei übernommen. Die erste Zeile der Datei muß die Arbeitsplannummer enthalten, die Reihenfolge der anderen Daten ist beliebig. Die Daten der Arbeitsplanpositionen werden durch die Textkürzel AVOBEGIN und AVOEND geklammert, die jeweils am Anfang einer eigenen Zeile stehen. Zwischen diesen Kennungen sind die Arbeitsplanpositionsdaten einzufügen. Es kann beliebig viele Arbeitsplanpositionen zu einem Arbeitsplan geben, die alle nach den Daten des Arbeitsplanes folgen müssen.

Triton Begriff	Pflicht	Format	Textkürzel	Bemerkung
Arbeitsplan	J	CHAR(13)	ARBEITSPLAN	
Artikelcode	J	CHAR(16)	HERST_ID	
Planer	J	CHAR(15)	ERSTELLER	
Übergabedatum	J	JJJJ.MM.DD: HH.MM.SS	ERSTELLDZ	
Bezeichnung	N	CHAR(60)	KOMMENTAR	

Tab. 3.3: Arbeitsplandaten der Triton-Schnittstelle

Arbeitsvorgangsdaten

Triton Begriff	Pflicht	Format	Textkürzel	Bemerkung
Position	J	CHAR(13)	APLINDEX	
Arbeitsgang	J	CHAR(15)	AVG_NR	
Bezeichnung Arbeitsgang	N	CHAR(60)	KOMMENTAR	
Stückzeit + Rüstzeit	J	DDDD:HH.MM.SS	STUECKZEIT	
Geplante Menge	J	INTEGER(10)	STUECK	Nur bei Übertragung eines Auftrags
	J	INTEGER(1)	GESPERRT	0: Arbeitsvorgang wird übersprungen 1: Arbeitsvorgang wird nicht übersprungen
AVO-Typ	J	CHAR(3)	AVGTYP	Bearbeitungsart
Meßfrequenz	N	INTEGER(10)	MESSFREQ	
Fertigungshilfs-mittel	N	CHAR(15)	NCNR	Nummer des verwendeten NC-Programms

Tab. 3.4: Arbeitsplanpositionsdaten der Triton-Schnittstelle

Einplanungstermine

Arbeitsplanpositionen können bei einer Auftragsübergabe mit den Planterminen des PPS-Systems versehen sein. Diese Plandaten sind in einem getrennten Abschnitt innerhalb des Arbeitsplandatenabschnitts anzugeben und werden durch die Textkürzel TERMINBEGIN und TERMINEND geklammert, die jeweils am Anfang einer eigenen Zeile stehen. Es gibt pro Arbeitsplanposition nur einen solchen Block. Die Plantermine sind immer nach den Arbeitsplanpositionsdaten und immer vor den Bearbeitungsplatzdaten zu den Arbeitsplanpositionen anzugeben.

Triton Begriff	Pflicht	Format	Textkürzel	Bemerkung
geplantes Startdatum	J	JJJJ.MM.DD: HH.MM.SS	ANFANG	Nur bei Übertragung eines Auftrags.
Frühestes Enddatum	J	JJJJ.MM.DD: HH.MM.SS	ENDEFRUEH	Nur bei Übertragung eines Auftrags

Tab. 3.5: Termindaten der Triton-Schnittstelle

Bearbeitungsplätze

Arbeitsplanpositionen werden zusammen mit den dazugehörigen Bearbeitungsplätzen übernommen die durch die Textkürzel PLATZBEGIN und PLATZEND geklammert werden und jeweils am Anfang einer eigenen Zeile stehen. Es gibt pro Arbeitsplanposition nur einen solchen Block. Zwischen den Textkürzeln kann es beliebig viele Bearbeitungsplätze pro Arbeitsplanposition geben.

Triton Begriff	Pflicht	Format	Textkürzel	Bemerkung
(Fertigungshilfs-mittel)	J	CHAR(16)	PLATZ	Möglicher Bearbeitungsplatz für die Arbeitsplanposition

Tab. 3.6: Bearbeitungsplatzdaten der Triton-Schnittstelle

Fertigungshilfsmittel

Angaben zu Fertigungshilfsmitteln werden mit den Arbeitsplanpositionen übergeben. Die Daten der Fertigungshilfsmittel werden durch die Textkürzel FHMBEGIN und FHMEND geklammert, die jeweils am Anfang einer eigenen Zeile stehen. Es kann beliebig viele so gekennzeichnete Fertigungshilfsmittel pro Arbeitsplanposition geben.

Triton Begriff	Pflicht	Format	Textkürzel	Bemerkung
Fertigungshilfs-mittelnummer	N	CHAR(8)	FHMNUMMER	
Fertigungshilfs-mittelart	N	CHAR(3)	FHMTYP	

Tab. 3.7: Fertigungshilfsmitteldaten der Triton-Schnittstelle

Schnittstellen der Leitstände

Die Fertigungsleitstände für die einzelnen Fertigungsinseln laufen jeweils auf einem dedizierten Rechner und besitzen eigene Datenverwaltungsmodule auf Dateibasis. Ein direkter Zugriff mit externen Programme ist darauf nicht möglich. Jedes Leitstandsystem stellt auf dem eigenen Rechner Datenein- und -ausgabeverzeichnisse zur Verfügung, über welche die Kommunikation abgewickelt werden muß. Die Kommunikation erfolgt mit sequentiellen Dateien, deren Namen je nach Objekt, über welches Daten ausgetauscht werden, eine unterschiedliche Endung besitzen. Sie müssen in ein Eingabeverzeichnis abgelegt werden, bzw. stehen in einem Ausgabeverzeichnis zur Verfügung. In diesen Dateien wird jedes Datum durch ein Textkürzel gekennzeichnet. Die Verzeichnisse werden in festen Zeitintervallen auf vorhandene Dateien überprüft. Nicht vorhandene Daten, die vom Fertigungsleitstand benötigt werden, können durch eine Anforderungsdatei im jeweiligen Ausgabeverzeichnis angefordert werden. Dieses sind insb. NC-Programme und FUD. Der Dateiaufbau ist durch die folgende Spezifikation festgelegt:

Fertigungsauftrag

Textkürzel	Datum	Format	Länge
FTGAUFTRAG:	Auftragsnummer	alphanumerisch	13
KNDAUFTRAG:	Nummer des Kundenauftrags	alphanumerisch	20
ERSTELLER:	Ersteller des Auftrags	alphanumerisch	15
KOMMENTAR:	Kommentar zum Auftrag	alphanumerisch	60
ARBEITSPLA:	Arbeitsplannummer zum Auftrag	alphanumerisch	13
STUECK....:	Stückzahl des Auftrags	numerisch	10
PRIORITAET:	Priorität des Auftrags: 1: höchste Priorität 9: niedrigste Priorität	numerisch (1-9)	1
TERMIN....:	Einzuhaltender Endtermin des Auftrags	jjjj:mm:tt hh:mm:ss	19

PLANFREIG.:	Die Planungsfreigabe gibt an, ob der Auftrag bei Simulationsplanungen berücksichtigt werden soll.	numerisch	1
FERTFREIG.:	Die Fertigungsfreigabe gibt an, ob der Auftrag aus einem Simulationsplan in einen aktiven Belegungsplan übernommen werden darf.	numerisch	1
BEZEICHNG.:	Bezeichnung des Auftrags	alphanumerisch	60
PROJEKT...:	Projekt, zu dem der Auftrag gehört	alphanumerisch	79
FTGSTUFE..:	Fertigungsstufe, eine ABB-spezifische Kennzahl für den Fertigungsauftrag	alphanumerisch	3
LIEFERUNG.:	Geplanter Anliefertermin des Ausgangsmaterials an die Fertigungsinsel	jjjj:mm:tt hh:mm:ss	19
HERST_ID..:	Herstell-ID des Teils, welches hergestellt werden soll.	alphanumerisch	20
ABLIEFORT.:	Ablieferort, an den das Ausgangsmaterial gebracht werden muß.	alphanumerisch	8

Tab. 3.8: Schnittstellenspezifikation für Fertigungsaufträge

Arbeitspläne

Textkürzel	Datum	Format	Länge
ARBEITSPLA:	Arbeitsplannummer	alphanumerisch	13
ERSTELLER.:	Ersteller des Arbeitsplanes	alphanumerisch	15
ERSTELLDZ.:	Erstelldatum und -zeit des Arbeitsplanes	jjjj:mm:tt hh:mm:ss	19

KOMMENTAR.:	Kommentar zum Arbeitsplan	alphanumerisch	60
HERST_ID..:	Herstell-ID des herzustellenden Teiles	alphanumerisch	20

Tab. 3.9: Schnittstellenspezifikation für Arbeitspläne

Arbeitsplanpositionen

Zusätzlich müssen in der jeweiligen Arbeitsplandatei zu einem Auftrag für jede Position die folgenden Daten angegeben werden:

Textkürzel	Datum	Format	Länge
APL_INDEX..:	Positionsnummer innerhalb des Arbeitsplans.	alphanumerisch	4
AVG_TYP...:	Arbeitsvorgangstyp.	alphanumerisch	3
AVG_NUMMER:	Nummer des Arbeitsvorgangs.	alphanumerisch	13
KOMMENTAR.:	Kommentar zur Arbeitsplanposition.	alphanumerisch	60
STUECKZEIT:	Geplante Dauer.	tttt:hh:mm:ss	13
STUECK....:	Anzahl der zu bearbeitenden Teile.	numerisch	10
GESPERRT..:	Falls gesetzt, wird diese Position bei der Fertigung übersprungen.	numerisch	1
AUTOMATIK.:	Wird gesetzt, falls es möglich ist, den Bearbeitungsbeginn ohne Einfluß des Bedieners vollautomatisch auszulösen.	numerisch	1
MESSH.....:	Häufigkeit der Prüfvorgänge.	numerisch	3

Tab. 3.10: Schnittstellenspezifikation für Arbeitsplanpositionen

Für jeden Bearbeitungsplatz, auf dem eine Arbeitsplanposition ausgeführt werden kann, muß ein Eintrag nach folgendem Format erfolgen:

Textkürzel	Datum	Format	Länge

PLATZ....:	Bearbeitungsplatz.	alphanumerisch	24

Tab. 3.11: Schnittstellenspezifikation für Bearbeitungsplatzdaten

Für jede für eine Arbeitsplanposition benötigte Datendatei, insb. für NC-Programme, muß ein Eintrag nach folgendem Format erfolgen:

Textkürzel	Datum	Format	Länge
BEA_TYP...:	Typ der Bearbeitung, der auch den Typ der Datendatei festlegt.	alphanumerisch	3
BEA_NUMMER:	Nummer, die angibt, welche Dateinummer oder NC-Programmnummer benötigt wird.	alphanumerisch	15
BEA_ART...:	Art der Bearbeitung.	alphanumerisch	3
VERW_ORT..:	System, in dem die benötigte Datei verwaltet wird.	alphanumerisch	1

Tab. 3.12: Schnittstellenspezifikation für Bearbeitungdatendateien

Für jedes NC-Programm benötigt das Leitstandssystem bestimmte Informationen, die über eine Datei mit folgendem Aufbau übermittelt werden müssen:

Textkürzel	Datum	Format	Länge
BEA_TYP...:	Typ der Bearbeitug.	alphanumerisch	3
BEA_NUMMER:	Nummer der Datei oder des Programms.	alphanumerisch	15
BEA_ART...:	Art der Bearbeitung	alphanumerisch	3
ERSTELLER.:	Ersteller des Programms (der Datei)	alphanumerisch	15
ERSTELLDZ.:	Datum und Zeit der Erstellung	jjjj:mm:tt hh:mm:ss	19
KOMMENTAR.:	Kommentar zum Programm (zur Datei)	alphanumerisch	60

LAUFDR....:	Laufdauer des NC-Programms.	tttt:hh:mm:ss	13
STUECK....:	Anzahl der Teile, für welche die Laufdauer angegeben wird.	numerisch	10
MODFREIG..:	Diese Freigabe gibt an, ob der Leitstand eine Änderung des Programms in der Maschinensteuerung oder im Leitstand selber zulassen darf.	numerisch	1
VERSION..:	Version des Programms (der Datei).	numerisch mit Nachkommastellen	xxx, yyy

Tab. 3.13: Schnittstellenspezifikation für Informationen zu NC-Programmen

Für jedes von einem NC-Programm verwendete Werkzeug, Vorrichtung und Aggregat muß eine Werkzeugeinsatzliste, Vorrichtungseinsatzliste bzw. Aggregateinsatzliste übergeben werden, in der das Werkzeug, die Vorrichtung, bzw. das Aggregat mit den Einsatzdauern und dem Bedarfstermin relativ zum Beginn der NC-Bearbeitung angegeben wird. Der Aufbau dieser Dateien ist sequentiell und die Daten sind mit einem Textkürzel identifiziert. Das verwendete NC-Programmiersystem ist in der Lage, diese Dateien den Schnittstellen des Leitstandsystems konform zu erzeugen, so daß der Dateiaufbau für das zu erstellende System nicht relevant ist, denn diese können ohne Änderungen übertragen werden.

Schnittstellen des NC-Progammiersystems

Beim Compilieren des Quellcodes für einen bestimmten Steuerungstypen wird der Code nach DIN 66025 in einer Datei abgelegt. Ein weiteres Programm spaltet diese Datei auf und erzeugt neben dem DIN-Code eine Datei mit Kopfdaten, eine Werkzeugeinsatzliste, eine Vorrichtungseinsatzliste, eine Aggregateeinsatzliste und eine Kommentardatei. Diese Dateien entsprechen den Anforderungen der Importschnittstelle zum Soflex-FLS. Die Dateien werden derzeit auf einem zentralen Dateiserver im Netzwerk abgelegt und von dort in die Datenimportverzeichnisse der Leitstandrechner kopiert. Das Kopieren wird durch ein Programm realisiert, welches ständig das Verzeichnis auf dem Dateiserver auf neue Dateien prüft und, sofern welche vorhanden sind, diese dann kopiert. Dieses Kopierprogramm wird auf einem dedizierten Rechner[137] ausgeführt.

[137] Dieser Rechner wird als Loopserver bezeichnet.

Netzwerk Infrastruktur

Das im Bereich PM verwendete Netzwerk verbindet alle genannten Systeme und ist im Fertigungsbereich ein Ethernet nach IEEE 802.3 Standard und in den fertigungsnahen Bereichen als ein Token-Ring-Netzwerk implementiert. Als Netzwerkprotokolle werden im Fertigungsbereich, insb. bei den Leitständen, TCP/IP eingesetzt, in den fertigungsnahen Bereichen wird das NetBIOS-Protokoll benutzt. Alle am Netzwerk beteiligten Rechner sind damit gegenseitig über Dateitransfers über Fileserver erreichbar.

Unter Berücksichtigung der beschriebenen Systemen wird ein Schnittstellensystem entwickelt, um diese zu verbinden. Es muß sich in die existierende Systemlandschaft mit ihrer Netwerkinfrastruktur und den bestehenden Schnittstellen einfügen. Zur Systementwicklung ist eine detailierte Untersuchung und Festlegung der fachlichen Anforderungen an das System notwendig. Dies ist die zweite Phase in den beschriebenen Vorgehensmodellen.

Anforderungsanalyse

Das Schnittstellensystem soll Aufgaben der Wahrung der Datenintegrität, der Datentransformation, der Datensynchronisation und der Datenübertragung übernehmen.[138] Es muß in der Lage sein, den Informationsfluß zwischen den Systemen ereignisabhängig zu steuern. Die Entwicklung wird durch die geringe Anzahl beteiligter Systeme sowie den hauptsächlich in einer Richtung laufenden Informationsfluß, nämlich vom PPS zum Fertigungsleitsystem, vereinfacht. Die Anforderungen an die zu speichernden Daten, die Steuerung des Informationsflusses und einzuhaltende Integritätsbedingungen werden konkretisiert und im Anschluß durch geeignete Modelle formalisiert.

Das Schnittstellensystem muß die vom PPS-System bereitgestellten Produktionsplanungsdaten lesen und in einer Datenbank speichern. Dies sind insb. Fertigungsaufträge und Arbeitspläne mit Arbeitsplanpositionen. Das Schnittstellensystem muß vom NC-Programmiersystem bereitgestellte NC-Programme lesen und in der Datenbank ablegen. Es muß in der Lage sein, fertigungsunterstützende Daten, FUD,derzeit sind dies Texte und Grafiken, einzulesen und zu speichern. Dem Benutzer muß die Möglichkeit gegeben werden, eine Zuordnung zwischen den NC-Programmen bzw. FUD und den Arbeitsplanpositionen herzustellen und in der Datenbank zu hinterlegen. Das Schnittstellensystem ist dann in der Lage, diese Daten

138 Vgl. Kapitel .

dem Leitstandsystem in der benötigten Form bereitzustellen. Das Schnittstellensystem muß Rückmeldungen des Leitstandsystems über den Status von Auftragspositionen und Fertigungsaufträgen entgegennehmen und ebenfalls in der Datenbank zur weiteren Auswertung speichern. Diese Rückmeldungsdaten müssen dem PPS-System zur Verfügung gestellt werden.

Das Schnittstellensystem soll ferner dazu dienen, eine rudimentäre Verwaltung für NC-Programme und FUD zur Verfügung zu stellen. Insbesondere soll eine Versionierung und Versionskontrolle dieser Daten möglich sein. Dieses ist derzeit in keinem der eingesetzten Systeme möglich.

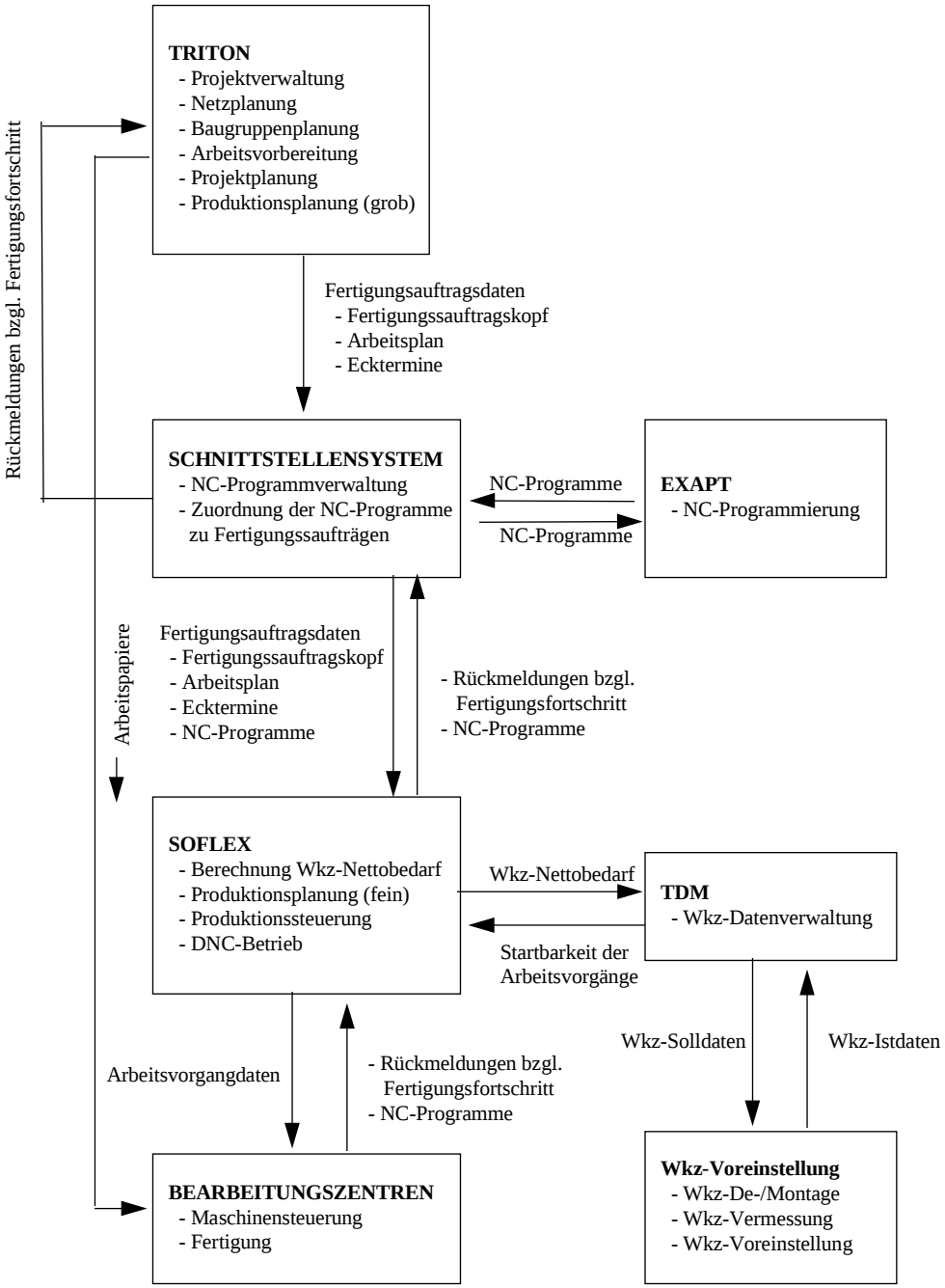


Abb. 3.1: Systemübersicht und Informationsfluß

Einlesen der Daten

Das Schnittstellensystem soll zum Einlesen der Daten des PPS-Systems ein einstellbares Dateiverzeichnis auf vorhandene Übergabedateien überprüfen. Findet es Dateien vor, so werden diese gelesen und anschließend gelöscht. Im Falle fehlerhafter Daten muß eine Fehlermeldung erzeugt werden und die Dateien sollen in ein anderes Verzeichnis kopiert werden. Das Einlesen von Daten des NC-Programmiersystems soll auf gleiche Weise geschehen.

Fertigungsaufträge

Fertigungsaufträge, bestehend aus dem Fertigungsauftragskopf und gekennzeichnet durch eine Fertigungsauftragsnummer, bezeichnen die Herstellung mehrerer Teile mit einer bestimmten Herstellteilidentifikationsnummer (Herstell-ID). Ein in das Schnittstellensystem eingelesener Fertigungsauftrag wird automatisch für die Simulationsplanung sowie die Belegungsplanung im FLS freigeben, da Fertigungsaufträge nur dann an das Schnittstellensystem übergeben werden, wenn sie innerhalb des PPS-Systems freigegeben sind

Dem FLS wird damit allerdings die Möglichkeit genommen, eine vorläufige Einplanung des Auftrages vorzunehmen. Vorstellbar ist eine Übergabe von Fertigungsaufträgen auch ohne Fertigungsfreigabe. Diese dürfen an das FLS zwar als planbar, allerdings nicht als fertigungsfreigegeben übertragen werden. Eine Änderung der Fertigungsaufträge und der dazugehörigen Arbeitspläne wäre in diesem Fall bis zum Zeitpunkt der Fertigungsfreigabe möglich. Hierdurch würde das FLS in die Lage versetzt, eine Simulationsplanung mit mehr oder weniger vollständigen Daten vorzunehmen.

Organisatorisch ist festzulegen, daß Produktionsaufträge die bereits an das Schnittstellensystem übertragen sind, weder in bezug auf die Arbeitsgangfolge noch in bezug auf Stückzahlen, geändert werden dürfen denn sie sind im PPS-System zur Fertigung freigegeben und eventuell bereits teilweise zurückgemeldet. Solche Änderungen müssen daher im Rahmen einer Änderung des Arbeitsplanes, d.h. durch eine Neuerstellung des Arbeitsplanes, durchgeführt werden. Der Fertigungsauftrag muß dann über eine Administrationskomponente aus dem Schnittstellensystem und ebenfalls aus dem FLS gelöscht werden können, um danach als neuer Fertigungsauftrag vom PPS-System, zusammen mit dem geänderten Arbeitsplan, neu übertragen zu werden. Der neu erstellte Fertigungsauftrag muß rückgemeldete Auftragspositionen und Stückzahlen bereits berücksichtigen.

Beim Übertragen der Fertigungsaufträge wird immer der aktuelle Arbeitsplan als Anhang an das System übertragen, um so die Aktualität der Daten sicherzustellen. Fertigungsaufträge werden nicht an das FLS übertragen, wenn der zugehörige Arbeitsplan gesperrt ist (vgl. Kapitel , Arbeitspläne). Der Fertigungsauftrag erhält dann nur den Status „fertigungsfreigegeben“, die Übertragung an das FLS muß zu einem späteren Zeitpunkt, nach Aufhebung der Sperrung im Arbeitsplan, manuell ausgelöst werden.

Eine Möglichkeit zum manuellen Anlegen von Fertigungsaufträgen und Arbeitsplänen, ohne das PPS-System einzubeziehen, soll nicht realisiert werden, um mögliche Inkonsistenzen zwischen PPS- und Schnittstellensystem auszuschließen. Es darf z.B. nicht vorkommen, daß eine Ressource durch einen manuell eingegebenen Auftrag belegt wird, wenn diese im PPS-System noch als verfügbar geführt wird. Diese Vorgehensweise ist allerdings noch innerhalb des FLS möglich, d.h. hier können Aufträge und Arbeitspläne manuell eingegeben werden. Dies muß durch Anweisungen an das Bedienpersonal ausgeschlossen werden.

Stückzahlen und Stückzeiten

Da im FLS keine Stücklisten geführt werden, besteht die Notwendigkeit, Informationen über die zu fertigende Werkstückanzahl im Arbeitsplan zu führen. Bei der Generierung des Fertigungsauftrages im PPS-System fügt dieses die aus der Stücklistenauflösung gewonnenen Stückzahlen in die Auftragspositionen des Fertigungsauftrages ein. Ebenso wird im Fertigungsauftragskopf die Anzahl der zu fertigenden Endprodukte festgelegt. In den Auftragspositionen sind also die für die zu fertigende Anzahl von Endprodukten benötigten Stückzahlen für die jeweiligen Auftragspositionen eingetragen.

Die Stückzeiten in den Auftragspositionen sind die Ausführungsdauern pro der in den Auftragspositionen angegebenen Stückzahlen. Das Soflex FLS berechnet die Gesamtdauer der Arbeitsvorgänge in folgender Weise: Stimmt die Anzahl der zu fertigenden Teile im Fertigungsauftrag mit der in der Auftragsposition angegebenen Stückzahl überein, so wird die Stückzeit als Gesamtdauer der Auftragsposition angenommen. Stimmen die beiden Stückzahlen nicht überein, so wird als Gesamtzeit die Stückzeit in der Auftragsposition, multipliziert mit der Stückzahl im Fertigungsauftrag, dividiert durch die Stückzahl in der Auftragsposition, angenommen.

Inselübergreifende Fertigungsaufträge

Inselübergreifende Fertigungsaufträge lassen sich über die in den Arbeitsplanpositionen angegebenen Bearbeitungsplätze und die zugeordnete Fertigungsinsel identifizieren. Inselübergreifende Fertigungsaufträge werden an alle an diesem Auftrag beteiligten Inseln gesendet. Die einzelnen Leitsysteme erhalten dabei aber nur die für sie relevanten Teile des Fertigungsauftrages.

Da auf PPS-Ebene eine Terminierung auch auf Auftragspositionsebene erfolgt, stehen für Auftragspositionen Ecktermindaten zur Verfügung. Mittels dieser Eckktermine muß den FLS mitgeteilt werden, wann die einzelnen Auftragsteilstücke in den Inseln fertiggestellt sein müssen. Dieses ist aufgrund des Belegungsalgorithmus des FLS notwendig: Das FLS priorisiert Fertigungsaufträge ausgehend vom spätest möglichen Anfangstermin, der aufgrund einer Rückwärtsrechnung vom spätest zulässigen Endtermin berechnet wird. Wenn Aufträge mehr als einmal in einer Insel bearbeitet werden müssen, muß dem FLS der Endtermin jedes Teilstückes mitgeteilt werden, da sonst die Bearbeitungszeiten in den anderen Inseln bei der Rückwärtsrechnung auf den spätest möglichen Starttermin nicht berücksichtigt werden können. Dieses würde zu einem späteren Starttermin führen, über den dann dem Fertigungsauftrag eine niedrigere Priorität zugewiesen würde als notwendig.

Den beteiligten FLS wird nicht mitgeteilt, wieviele Arbeitsschritte und wieviel Zeit das Werkstück in einer anderen Insel verbringen wird. Da das FLS auch keine Starttermine von Fertigungsaufträgen kennt, sondern alle Fertigungsaufträge so behandelt, als wären sie sofort startbar, kann nur eine sehr schlechte Belegungs- und Kapazitätsplanung durchgeführt werden. Auftragspositionen, die aufgrund der fehlenden Startterminrestriktion zu früh eingeplant werden, werden bei nicht Verfügbarkeit des Materials aufgeschoben. Es ist wünschenswert, in diesem Fall einen häufigen Neuaufwurf der Belegungsplanung durchzuführen.

Aber auch wenn nicht inselrelevante Auftragspositionen übertragen würden und auf einen virtuellen Bearbeitungsplatz gebucht werden, wird der Belegungsplan nur sehr beschränkt verbessert, denn es ist zu beachten, daß dies keine genauen Zeiten liefern kann; nur die reinen Bearbeitungs- und Rüstzeiten in der externen Insel sind bekannt, nicht aber die gesamten Warte- oder Durchlaufzeiten in dieser Fertigungsinsel.

Bei der Entscheidung, welche Teile eines Fertigungsauftrages an ein bestimmtes Leitsystem zu übergeben sind, spielt zum einen der zu jeder Auftragsposition zu übergebende Bearbeitungsplatz, der eindeutig einer Fertigungsinsel zugeordnet ist, eine Rolle. Zum anderen kann aber auf Grund von technischen Restriktionen auf einem

bestimmten Bearbeitungsplatz nur eine eingeschränkte Menge von Arbeitsvorgangsarten ausgeführt werden. Es müssen somit alle übergebenen Bearbeitungsplätze eindeutig auf eine Fertigungsinsel verweisen und die für die Auftragsposition angegebene Arbeitsvorgangsart muß auf allen angegebenen Plätzen möglich sein, so daß auch hierüber der Bezug zur Fertigungsinsel hergestellt werden kann.

Vom FLS wird gefordert, daß zu allen Auftragspositionen mindestens ein Bearbeitungsplatz angegeben wird. Problematisch sind dabei Positionen im Auftrag, die keine Bearbeitung, sondern eine Information des Werkers bedingen sollen. Da diese Verwendung des Arbeitsplanes im FLS nicht verloren gehen soll, muß ein virtueller Bearbeitungsplatz geschaffen werden, auf dem solche Auftragspositionen ausgeführt werden können und somit dem FLS bekanntgemacht werden. Ebenso muß hierzu ein virtueller Arbeitsvorgangstyp definiert sein.

Arbeitspläne

Ein Arbeitsplan ist gekennzeichnet durch die Angabe der Herstell-ID des Teiles, dessen Herstellung er beschreibt, sowie die Angabe der Arbeitsplannummer. Arbeitspläne können im Schnittstellensystem ohne zugehörigen Fertigungsauftrag gespeichert werden, nicht jedoch Fertigungsaufträge ohne Arbeitspläne. Ein evtl. vorhandener Arbeitsplan mit derselben Herstell-ID und Arbeitsplannummer wird nicht überschrieben, sondern erhält den Status eines Backups. Ein eventuell vorhandener Backup-Arbeitsplan wird mitsamt den Arbeitsplanpositionen gelöscht, sobald ein neuer Arbeitsplan mit derselben Herstell-ID und Arbeitsplannummer eingelesen wird. Es befinden sich also maximal zwei Versionen desselben Arbeitsplanes mit dazugehörigen Arbeitsvorgängen im System. Dies ist nicht nur für eine fehlertolerante Bedienung des Systems, um z.B. ungewolltes Überschreiben zu verhindern, wünschenswert. Dem Bediener soll die Möglichkeit gegeben werden, bereits während der Arbeitsplanung und NC-Programmierung eine Zuordnung zu den Arbeitsplanpositionen vorzunehmen. Diese sollen dann bei Arbeitsplanänderung soweit wie möglich übernommen werden. Das Schnittstellensystem soll also auch als Kommunikationsmittel zwischen Arbeitsplanung und NC-Programmierung dienen und die Kooperation unterstützen, evtl. bis zur Form eines Auftragsteams, welches die integrierte Bearbeitung eines Fertigungsauftrags durchführen kann.

Arbeitspläne können im Schnittstellensystem gesperrt werden. Ein gesperrter Arbeitsplan kann nicht für eine Planung innerhalb des FLS verwendet werden und darf dementsprechend nicht an das FLS übertragen werden (vgl. Kapitel). Arbeitspläne mit

gesperrten Position dürfen aber an das FLS übertragen werden. Der Arbeitsvorgang wird im FLS bei der Fertigung übersprungen.

Eine Entsperrung des Arbeitsplanes ist manuell möglich, falls eine Verfügbarkeitsprüfung der zugeordneten NC-Programme und FUD dieses zuläßt. Ein gesperrter Arbeitsplan darf, unabhänging von anderen Kriterien, nicht an das FLS übergeben werden. Der Freigabestatus eines neu eingelesenen Arbeitsplanes setzt sich aus der Verfügbarkeit aller den Arbeitsplanpositionen zugeordneten NC-Programmen und FUD und zum anderen aus dem Freigabestatus des Vorgängers zusammen. Sind z.B. alle zugeordneten NC-Programme und FUD vorhanden, war der Vorgänger aber gesperrt, so bleibt der neue Arbeitsplan ebenfalls gesperrt. Nur wenn alle zugeordneten NC-Programme und FUD tatsächlich vorhanden sind, und der Vorgänger nicht gesperrt war, wird der neu eingelesene Arbeitsplan entsperrt.

Durch eine explizite Sperrmöglichkeit des Arbeitsplanes ist auch die Möglichkeit gegeben, NC-Programm- und FUD-Zuordnungen vorzunehmen, diese bereitzustellen und den Arbeitsplan dennoch nicht freizugeben, sei es, weil noch nicht alle Zuordnungen erfolgt sind oder weil Fehler festgestellt wurden.

Arbeitspläne dürfen ebensowenig wie Fertigungsaufträge geändert werden, wenn sie bereits an das FLS übertragen wurden, da eventuell vorhandene Rückmeldungen auf Positionen des Arbeitsplanes referenzieren.[139]

Arbeitsplanpositionen

Die zu einem Arbeitsplan gehörenden Arbeitsplanpositionen sind gekennzeichnet durch die Identifikation des zugehörigen Arbeitsplanes, Arbeitsplannummer und Herstell-ID, sowie den Arbeitsplanindex. Innerhalb jeder Arbeitsplanposition wird angegeben, auf welcher Maschine oder Maschinengruppe, diese Position ausführbar ist. Die zu einem Arbeitsplan gehörenden Arbeitsplanpositionen werden ebenfalls als Backup geführt.

Ein Arbeitsvorgang kann innerhalb des Systems nicht explizit gesperrt werden. Eine implizite Sperrung ist abhängig von der Verfügbarkeit der zugeordneten NC-Programme und FUD. Die Kennung „Gesperrt“ im Arbeitsvorgang ist ein internes Merkmal für das FLS und ist nicht von der Verfügbarkeit der zugeordneten FUD abhängig. Im folgenden soll unter der Verfügbarkeit der NC-Programme und FUD nicht nur das Vorhandensein in der Datenbank des Schnittstellensystems, sondern auch die Freigabe verstanden werden.

---

139 Vgl. Kapitel

Beim Einlesen der Arbeitsplanpositionen wird wie folgt verfahren. Sind von der Arbeitsplanposition benötigte NC-Programme und FUD bereits angegeben, so wird überprüft, ob diese dem Schnittstellensystem in der Datenbank zur Verfügung stehen. Ist dies der Fall, so wird die Arbeitsplanposition als implizit freigegeben gespeichert, anderenfalls ist die Arbeitsplanposition als gesperrt zu betrachten. Die angegebenen FUD-Zuordnungen werden aber auch gespeichert, wenn die NC-Programme und FUD dem Schnittstellensystem nicht zur Verfügung stehen. Es werden dann Versionen mit der Versionsnummer 0 angelegt, die keine Daten enthalten. Sind keine NC-Programme oder FUD angegeben, so wird geprüft, ob in der Backup-Version des Arbeitsvorganges bereits Zuordnungen vorhanden waren, die dann übernommen werden. Durch die Übernahme der Zuordnungen wird nicht nur ermöglicht, daß ein freigegebener Arbeitsplan ohne weitere manuelle Bearbeitung ersetzt werden kann und damit ein vollautomischer Betrieb durchgeführt werden kann, sondern auch, daß dem Benutzer eine erneute, zeitaufwendige Zuordnung erspart wird wenn sich ein Arbeitsplan geringfügig ändert.

Datenänderungen

Da bereits freigegebene Fertigungsaufträge im PPS-System nicht mehr geändert werden dürfen, kann es nicht vorkommen, daß ein Fertigungsauftrag mit einem Arbeitsplan, die zusammen an ein FLS übertragen wurden, durch eine neuere Version ersetzt werden muß. Auch das Übertragen eines einzelnen Arbeitsplanes durch das PPS muß verhindert werden wenn sich dieser Arbeitsplan bereits mit einem dazugehörigen Fertigungsauftrag bereits in einem FLS befinden. Eventuell eingelesene Fertigungsaufträge und Arbeitspläne werden in einer solchen Situation ignoriert. Stellt sich heraus, daß in der Fertigung aufgrund von Fehlern im Arbeitsplan oder Auftrag ein falsches Erzeugnis oder das Erzeugnis falsch hergestellt wird, so muß der Auftrag über eine Administrationskomponente manuell mit dem dazugehörigen Arbeitsplan aus dem Schnittstellensystem gelöscht werden. Ebenso muß der Auftrag dann aus dem FLS gelöscht werden. Danach muß der korrigierte Auftrag, eventuell um die bereits fertiggemeldeten Teile gekürzt, neu übertragen und freigegeben werden.

Da Fertigungsaufträge aber immer mit den dazugehörigen Arbeitsplänen übertragen werden, kann sich eine Situation ergeben, in der ein Fertigungsauftrag bereits mit einem Arbeitsplan an ein FLS übertragen wurde, jetzt aber ein neuer Fertigungsauftrag mit diesem Arbeitsplan ausgelöst wird. In dieser Situation wird der Fertigungsauftrag eingelesen, nicht aber der Arbeitsplan, denn es befindet sich bereits ein gültiger und freigegebener Arbeitsplan im System. Da eine Änderung der Arbeitspläne von Fertigungsaufträgen die bereits in der Fertigung sind auch im PPS-System nicht

möglich ist, ist der Arbeitsplan in diesem Fall der gleiche, d.h. es tritt nicht das Problem auf, daß ein Fertigungsauftrag zwar mit einem bestimmten Arbeitsplan freigegeben wird, aber dennoch mit einem anderen gefertigt wird.

NC-Programme und fertigungsunterstützende Daten

Fertigungsunterstützende Daten (FUD) sind derzeit Textdateien und Grafikdateien. Sie beschreiben benötigte Fertigungshilfsmittel oder dienen selbst zur Unterstützung der Fertigung. NC-Programme werden mit dem Programmiersystem EXAPT erzeugt. Ein Postprozessor wandelt den Quellcode in maschinenlesbaren DIN 66025 Code um und erzeugt eine Datei mit allgemeinen Informationen zum Programm, eine Aggregateeinsatzliste, eine Vorrichtungseinsatzliste, eine Werkzeugeinsatzliste und eine kommentierende Textdatei. Diese Dateien werden zusammen mit der DIN-Code-Datei in die Datenbank des Schnittstellensystems eingelesen. Ein Einlesen findet nur statt wenn alle Dateien zum Zeitpunkt des Einlesens zusammen vorhanden sind. Dazu müssen sie den gleichen Dateinamen mit einer für den Datentyp spezifischen Endung besitzen. Im folgenden wird unter NC-Programm die Einheit aus diesen Dateien verstanden. Bereits vorhandene Versionen von eingelesenen NC-Programmen und FUD werden nicht überschrieben, sondern bekommen eine neue Versionsnummer zugewiesen. Versionsnummern sind Fließkommazahlen, wobei die Vorkommastellen die Hauptversion bezeichnen, d.h. die Anzahl der Änderungen, die der NC-Programmierer/Bearbeiter durchgeführt hat. Die Nachkommstellen geben an, wie oft die Daten bereits im FLS oder, im Fall von NC-Programmen, der Maschinensteuerung geändert wurden. Die Möglichkeit, Änderungen direkt an der Maschine durchführen zu können, ist notwendig, da Simulationsverfahren zwar helfen können Kollisionen (des Werkzeugs mit dem Werkstück, ungewollt) und andere Fehler in NC-Programmen aufzudecken, die Erfahrung des Maschinenbedieners aber im Hinblick auf Optimierung des Programmes nicht unbeachtet bleiben soll. Jedem NC-Programm und FUD wird eine aktuelle Version zugeordnet. Die aktuelle Version gibt an, welches die derzeit gültige Ausfertigung ist, d.h. diejenige die bei der nächsten Anforderung an das FLS geschickt wird.

NC-Programm- und FUD-Versionen, aber nicht die aktuellen Versionen, können im Schnittstellensystem gesperrt werden. Wenn ein geändertes NC-Programm vom FLS rückübertragen wird, wird nur die rückübertragene Version gesperrt. Hier muß dann der NC-Programmierer einen manuellen Abgleich durchführen und die neue Version eventuell zur aktuellen erklären. Überträgt der NC-Programmierer eine neue Version, wird diese nicht automatisch zur aktuellen Version erklärt. Die Freigabekennung, die der NC-Programmierer in der Informationsdatei angibt, wird übernommen. Eine

Rückübertragung von NC-Programmen vom FLS kann ebenfalls nur dann stattfinden, wenn alle sechs zu diesem Programm gehörigen Dateien gleichzeitig vorhanden sind.

Ändern und Löschen von fertigungunterstützenden Daten

Existieren zu einem NC-Programm oder einer FUD-Datei Auftragspositionen, die bereits an das FLS übertragen wurden, dürfen diese Bearbeitungsdaten nicht gelöscht werden. Es darf aber eine neue Version erstellt werden und die aktuelle Version darf umgestellt werden, da das FLS keine bestimmte Version anfordert. Die aktuelle Version darf nicht gesperrt werden, denn dieses würde eine Datenübertragung auf FLS-Anforderung verhindern, bzw. bedeuten, daß das FLS die Fertigung bereits mit falschen Daten durchführt. In diesem Fall muß über eine Administrationskomponente der Auftrag manuell aus dem Schnittstellensystem gelöscht werden und dann neu übertragen werden. Hierbei muß die Stückzahl um bereits fertiggemeldete Teile reduziert werden.

Datenübertragung an die Leitstandsysteme

An ein FLS werden immer Fertigungsauftragsköpfe mit über den Arbeitsplan zugeordneten Auftragspositionen übergeben. Ein FLS erhält also keine neutralen Arbeitspläne, sondern nur fertigungsauftragsbezogene Arbeitspläne. Eine Übertragung von Daten kann durch mehrere Ereignisse ausgelöst werden.

Einem FLS wird ein Arbeitsplan übergeben, wenn das PPS-System einen Fertigungsauftrag übermittelt hat, zu dem bereits ein nicht gesperrter Arbeitsplan vorliegt. Dieser Auftrag ist dann für die Planung im Rahmen einer Simulationsplanung oder einer Belegungsplanung freigegeben und es darf mit der Fertigung begonnen werden. Sobald Aufträge an ein FLS übertragen werden, wird zu jeder darin enthaltenen Auftragsposition ein Status geführt, welcher den aktuellen Status des Arbeitsvorganges aufnimmt.

Ferner muß eine Datenübertragung stattfinden, wenn ein FLS NC-Programme oder FUD anfordert. Diese Daten müssen in der jeweils aktuellen Version dem FLS zur Verfügung gestellt werden und dürfen daher nicht gesperrt sein. Dieses wird bereits sichergestellt, wenn der freigegebene Auftrag an ein FLS übertragen wird. Im DNC-Betrieb muß die Übermittlung von NC-Programmen in der aktuellen Version manuell ausgelöst werden.

Rückmeldungen

Rückmeldungen des FLS werden eingelesen und der jeweiligen Auftragsposition zugeordnet. Eine Auftragspositionsmeldung ist gekennzeichnet durch eine bestimmten Arbeitsplanposition eines bestimmten Arbeitsplanes eines bestimmten Fertigungsauftrages sowie dem aktuellen Status dieser Auftragsposition. Mitgeführt wird weiterhin das Datum und die Uhrzeit der Statusänderung. Das FLS erzeugt Rückmeldungen entweder in festen zeitlichen Abständen oder ereignisorientiert, z.B. bei Abschluß eines Arbeitsganges. Einer Auftragsmeldung kann ein bestimmter Status nur einmal zugewiesen werden. Es wird eine Historie über alle Statusänderungen mit Datum und Uhrzeit erhalten, so daß nachträgliche statistische Auswertungen über Bearbeitungsdauern und Durchlaufzeiten möglich sind. Rückmeldungen werden vom FLS nicht nur für Arbeitsvorgänge erzeugt, sondern auch für den gesamten Fertigungsauftrag. Diese Rückmeldungen werden im Fertigungsauftrag selbst gespeichert. Auftragspositionen erhalten standardmäßig den Status 0. Dieser gibt an, daß der Arbeitsvorgang einem Fertigungsauftrag im Schnittstellensystem zugeordnet wurde. Der Status 9 gibt an, daß die Arbeitsplanposition im Rahmen einer Übertragung eines Fertigungsauftrages an ein FLS übergeben wurde. Die Status 1-3 entsprechen direkt den in den Rückmeldungen enthaltenen FLS-Status in der Form wie sie vom FLS gemeldet werden.

Mehrmandantenfähigkeit

Durch die Übergabe des Mandanten, durch den der Fertigungsauftrag erzeugt wurde, durch das PPS-System soll sichergestellt werden, daß Rückmeldungen die von dem Schnittstellensystem an das PPS-System übergeben werden sollen, an den korrekten Mandanten innerhalb des PPS-Systems gelangen. Der Mandant wird nicht als Ersteller des Fertigungsauftrages oder Arbeitsplanes angegeben, da diese im PPS-System als Benutzer namentlich bekannt sind. Die übergebene Mandantenkennung wird nicht an das FLS weiterübertragen, da dieses nicht mehrmandantenfähig ist.

Fachkonzept des Schnittstellensystems

Nachdem die fachlichen Anforderungen in verständlicher Formulierung festgehalten sind, muß nun eine Formalisierung dieser Anforderungen mit geeigneten Methoden und Modellen im Rahmen der Erstellung des Fachkonzepts folgen.

Datensicht

Ausgehend von den in Kapitel genannten Anforderungen an das Schnittstellensystem wird mit Hilfe des CASE-Werkzeugs Designer/2000 ein Entity-Relationship-Modell erstellt. Dieses ist in Anhang als Diagramm dargestellt. Zusätzlich zu den Entity-Sets sind deren Attribute und alle Relationships zwischen den Entity-Sets dargestetellt. Soweit durch die Konstrukte der Enitity-Relationship-Modellierung abbildbar, werden alle gestellten Anforderungen an die Datenintegrität berücksichtigt. Die Attribute und deren Datentypen orientieren sich an den Erfordernissen der Schnittstelle des Soflex-Systems und sind in Anhang aufgeführt.

Zentrales Objekt des Modells ist die Arbeitsplanposition, dargestellt durch das Entity-Set AVO. Es ist existentiell abhängig vom Arbeitsplan, APL, und ist innerhalb dieses gekennzeichnet durch die Angabe der Positionsnummer oder Index, APL_INDEX. Zusätzlich enthält es Attribute für die Aufnahme eines Kommentars, der Bearbeitungsdauer und der Anzahl der zu fertigenden Stücke[140], der Meßfrequenz, eines Sperrkennzeichens etc. Der Arbeitsplan, dargestellt durch das Entity-Set APL, enthält neben den identifizierenden Attributen Arbeitsplannummer, ARBEITSPLA, Herstell-ID des Teiles dessen Herstellung er beschreibt, HERST_ID, und dem Backup-Kennzeichen, BAK, welches die Werte 0 oder 1 annehmen kann, Attribute zur Aufnahme eines Kommentars, des Erstelldatums und weiterer Kennzeichen. Durch das Backup-Kennzeichen, BAK, ist es möglich, zu einem Herstellteil mehrere Arbeitpläne bzw. zu einem Arbeitsplan und damit zu allen Arbeitsplanpositionen und deren Zuordnungen zu Bearbeitungsplätzen etc, die aktuelle und die vergangene Version zu verwalten. Durch die Existenzabhängigkeit der Arbeitsplanposition vom Arbeitsplan werden dessen Schlüsselattribute auch auf die Arbeitsplanposition übertragen.

Fertigungsaufträge, AUFTRAG, sind gekennzeichnet durch die Fertigungsauftragsnummer, FTGAUFTRAG. Weitere Attribute dienen zur Aufnahme des aktuellen Auftragsstatus, STATUS, der expliziten Prioritätsangabe für den Auftrag, PRIORITAET, den Planungs- und Fertigungsfreigaben, PLANFREIG und FERTFREIG, der Anzahl der herzustellenden Stücke, STUECK, der Fertigungsstufe, FTGSTUFE, des Ablieferortes des Materials, ABLIEFERORT, des auslösenden Mandanten des PPS-Systems, MANDANT, u.a. Das Entity-Set enthält Attribute für die Aufnahme der rückgemeldeten Stückzahl, der Priorität die durch das Leitsystem vergeben wurde, PRIO_IST, und den Freigaben, PLFREI_IST, FEFREI_IST, die evtl. im Leitsystem geändert wurden.

[140] Vgl. dazu die Stückzeitproblematik in Kapitel .

Fertigungsaufträge beziehen sich auf einen Arbeitsplan. Durch diesen Bezug werden auch die Positionen des Arbeitsplanes mit dem Auftrag verknüpft, dargestellt durch das Entity-Set VIB („Vorgang in Bearbeitung“), welches als identifizierende Attribute die des Auftrages und die der Arbeitsplanposition übernimmt. Dieses Entity-Set nimmt den aktuellen Status der Auftragsposition aus den Rückmeldungen der Leitstandsysteme auf. Eine Arbeitsplanposition kann nur ein einziges Mal einen bestimmten Status erhalten. Es ist also nicht möglich, z.B. den Status „Fertig“ zweimal zu erreichen. Die Rückmeldungen auf Auftragsebene werden in Attributen des Entity-Set AUFTRAG, z.B. STUECK_IST, PRIO_IST, abgelegt. Einer ebensolchen Beziehung zwischen Auftrag und Arbeitsplanpositione lassen sich auch die Plantermine, PLANUNGSTERMIN, zuordnen. Während jedoch die VIB im Zeitverlauf mehrere Rückmeldungen aufnehmen müssen, werden die Plantermine nur einmal zugeordnet, sind daher als separate Entity-Sets modelliert.

Die Fertigungsinseln, konkret, die Soflex-Leitsysteme der Fertigungsinseln, dargestellt durch das Entity-Set SOFLINSEL, sind gekennzeichnet durch die Bezeichnung der Fertigungsinsel, SOFLINSEL, und besitzen als Attribute die jeweiligen Dateiübergabeverzeichnisse, INPFAD und OUTPFAD, über die die Kommunikation abgewickelt werden kann.

Jeder Insel ist eine Menge von Bearbeitungsplätzen, BEARBEITUNGSPLATZ, zugeordnet, die in den beiden beteiligten Systemen unterschiedlich benannt sind. Sie sind daher intern numeriert, PLATZNR, und nach außen mit beiden betrieblich gebräuchlichen Bezeichnungen, SOFLEXNAME, TRITONNAME, sichtbar. Auf einem Bearbeitungsplatz kann eine Menge von Arbeitsvorgangstypen, AVGTYP, ausgeführt werden, z.B. Drehen, Fräsen, etc. Diese orientieren sich an den durch das PPS-System vergebenen Typen. Jeder Arbeitsplanposition ist ein solcher Arbeitsvorgangstyp und dadurch eine Menge von Bearbeitungsplätzen zugeordnet, auf denen die Ausführung technisch möglich ist. Da bei der Arbeitsplanung für jede Arbeitsplanposition durch den Arbeitsplaner eine Menge möglicher alternativer Bearbeitungsplätze angegeben wird, ist eine Plausibilitätsprüfung bezüglich der Ausführbarkeit einer Arbeitsplanposition auf einem Bearbeitungsplatz möglich.

Die Bearbeitungsstationen, die Maschinen, sind durch das Entity-Set STATION repräsentiert und gekennzeichnet durch deren Bezeichnung, STATION. Eine Station kann auf einer Menge von Bearbeitungsplätzen arbeiten, umgekehrt ist ein Bearbeitungsplatz von einer Menge von Stationen aus erreichbar.[141] Für die Stationen ist

[141] Z.B. sind die Gantries der Doppel-Gantry-Anlagen in der Großmechanik über alle Plattenfelder und Planscheiben verfahrbar, auch über die, welche normalerweise der anderen Gantry zugeordnet sein sollten.

wiederum eine Menge von NC-Programmversionen, NCNRVER, freigegeben, STATFREIG. Diese Freigaben werden unterschieden in Start- und Übertragungsfreigaben (STARTFREI, TRANSFREIG).

Eine NC-Programmversion, NCNRVER, innerhalb einer NC-Programmnummer identifiziert durch die Versionsnummer, VER, enthält weitere Attribute für einen Kommentar zur Version, ein Sperr-Kennzeichen, GESPERRT, welches vom Programmierer gesetzt werden kann, um die Übertragung der Programmversion an das Leitsystem zu verhindern, sowie Angaben zur Laufdauer, LAUFDR, und der Anzahl der Werkstücke, auf die sich diese bezieht, STUECK. Daneben kann eine Veränderung durch das Leitsystem durch das Setzen der Modifizierungsfreigabe, MODFREIG, erlaubt oder verhindert werden..

Eine NC-Programmversion besteht neben dem DIN-Code aus weiteren Datenblöcken. Die zulässigen NC-Bearbeitungsdatenarten für diese Datenblöcke, derzeit die Werkzeugeinsatzliste, Vorrichtungseinsatzliste etc., sind durch das Entity-Set BEAART mit dem identifizierenden Attribut Datentypkennung, DTK, beschrieben. Jeder NC-Bearbeitungsdatenart ist ein Editor, EDITOR, zugeordnet, mit dem ein Datum des jeweiligen Typs bearbeitet werden kann. Die Daten selbst werden durch das Entity-Set NC-Bearbeitungsdaten, BEADAT, mit dem einzigen Attribut DATUM, welches eben diese Daten als Binärobjekte aufnimmt, dargestellt. Jeder Arbeitsplanposition kann genau ein NC-Programm, gekennzeichnet durch die NC-Programmnummer NCNR des Entity-Sets NCNR, zugeordnet werden. Beim Übertragen der Daten zum Leitstand wird die aktuelle Version, AKTVER im Entity-Set NCNRVER, ermittelt und nur diese wird übertragen.

Eine ähnliche Situation wie bei den NC-Programmen liegt bei den fertigungsunterstützenden Daten (FUD), derzeit Texte und Grafiken, vor. Sie werden durch das Entity-Set BEAVER dargestellt und sind gekennzeichnet durch die Versionsnummer VER. Weitere Attribute sind neben den Daten selbst, DATEN, auch Kommentar und Informationen zu Ersteller und Erstalldatum, ERSTELLER, ERSTELLDZ. Auch für die FUD ist jeweils eine der Versionen die aktuelle, gekennzeichnet durch das Setzen des Attributs AKTVER.

Jedes dieser FUD, BEARB, besitzt eine Nummer, NR, und ist gekennzeichnet durch einen Datentyp, dargestellt durch das Entity-Set TYP mit dem Datentyp, TYP, dem jeweils ein Editor, EDITOR, zugeordnet ist, um solche Daten bearbeiten zu können. Im Gegensatz zu NC-Programmen können einer Arbeitsplanposition jedoch beliebig viele solcher FUD zugeordnet werden. Bei einer Datenübertragung an die Leitstandsysteme wird nur die aktuelle Version übertragen.

Eine weitere Plausibilitätsprüfung wird durch den Ringschluß von Arbeitsplanpositionen, über NC-Programme, über freigegebene Stationen und deren Bearbeitungsplätze möglich.

Steuerungssicht

Die Steuerungssicht auf das Schnittstellensystem wird mit Hilfe des Werkzeugs INCOME der Firma Promatis als Petri-Netz modelliert. Der Grobablauf der Planung ist aus dem Petri-Netz in Anhang zu entnehmen.

Verkaufsangebote werden mit Wahrscheinlichkeiten für den Vertragsabschluß gewichtet. Sie gehen als gewichtete Verkaufsangebote in die grobe Auslastungsplanung ein, die den langfristigen Produktionsplan aktualisiert. Trifft zu einem gewichteten Angebot eine Bestellung ein, so wird das Angebot in einen Kundenauftrag überführt, der einer groben terminlichen und technischen Machbarkeitsprüfung unterzogen wird. Dies geschieht unter Berücksichtigung des langfristigen Produktionsplanes. Ausgehend von der technischen und terminlichen Machbarkeit kann der Auftrag abgelehnt werden, konzernintern an andere Standorte vergeben werden, oder aber angenommen werden. Bei einer internen Vergabe und einer Annahme wird der langfristige Produktionsplan entsprechend aktualisiert. Durch die Auftragsannahme wird die Erstellung von Konstruktions- und Fertigungszeichnungen ausgelöst und die Grobtermine der Hauptkomponenten werden abgestimmt. Danach kann eine Grobkalkulation durchgeführt werden, nach deren Abschluß das Projekt in Teilprojekte für die Bereiche KW/PM, KW/PT und KW/PG strukturiert wird. Für diese Projekte werden Aktivitäten ermittelt, diese werden zu einem Aktivitätennetz verknüpft und anschließend zu Baugruppen zugeordnet. Diese werden zum Großteil aus projektneutralen Daten kopiert und dann projektspezifisch ergänzt. Nachdem Aktivitäten und Baugruppen so festgelegt sind, kann das Projekt in den Bereichen freigegeben werden. Die Freigabe löst die Überprüfung der Teilestammdaten aus. Diese kann dazu führen, daß Teilestammdaten unter Zuhilfenahme von Zeichnungen geändert werden müssen. Nach der Freigabe der Projekte wird auch die Überprüfung der Stücklisten und Arbeitspläne ausgelöst. Je nach Ergebnis dieser Überprüfung müssen Stücklisten evtl. erstellt oder verändert werden. Dieses geschieht wiederum unter Zuhilfenahme von Zeichnungen. Auch Arbeitspläne werden je nach Prüfergebnis neu angelegt oder vervollständigt. Die Neuanlage von Arbeitsplänen geschieht durch das Hinzuziehen von vorhanden Arbeitsplänen aus dem projektneutralen Datenbereich. Ist ein Arbeitsplan nicht vollständig, so müssen Arbeitsplanpositionen, NC-Programme und deren Zuordnungen erzeugt werden. Diese müssen sich auf vorhandene Arbeitsvorgänge beziehen und durch diesen Bezug werden sie auch den Abteilungen, Maschinen und Bearbeitungsplätzen zugeordnet. Nachdem

die NC-Programmnummern für eine bestimmte Arbeitsplanposition vergeben sind, muß das NC-Programm programmiert werden. Der so entstehende Quellcode wird unter Bezugnahme auf den Bearbeitungsplatz der zugeordneten Arbeitsplanposition maschinenspezifisch compiliert.

Nachdem Teilestammdaten, Stücklisten und Arbeitspläne vervollständigt sind, kann eine Aktualisierung der Projektaktivitäten bezügl. der Kapazitätsbeanspruchung und zeitlichen Abhängigkeit erfolgen. Daraufhin erfolgt eine Netzplanung. Hierdurch entsteht ein Entscheidungsbedarf bezüglich der Fremdvergabe von Komponten aufgrund kapazitiver Restriktionen. Dieses wird in den Teilestammdaten hinterlegt. Durch die Netzplanung werden Termine für die Baugruppenfertigung und der Kapazitätsbedarf ermittelt. Unter Berücksichtigung der aktuellen Kapazitätsbelastung und des Kapazitätsangebotes der Abteilungen und der Maschinen wird manuell ein Kapazitätsabgleich durchgeführt. Danach wird ein PRP-Lauf durchgeführt. In diesen gehen neben den Terminen für die Baugruppen und der aktuellen Kapazitätsbelastung auch die Lagerbestände, die Kapazitätsangebote der Abteilungen und Maschinen, die Arbeitspläne mit Arbeitsplanpositionen und deren Zuordnungen zu den in Anspruch genommenen Abteilungen und Maschinen ein. Durch den PRP-Lauf werden Lagerauftrags-, Bestell- und Fertigungsauftragsvorschläge erzeugt, die, wiederum unter Einbeziehung der genannten Faktoren, manuell mit den Kapazitätsangeboten der Maschinen und Abteilungen abgeglichen weden. Nachdem dies geschehen ist, werden die Vorschläge bestätigt und in konkrete Lageraufträge, Bestellungen und Fertigungsaufträge mit Terminen überführt. Bestellungen werden versandt und so in laufende Bestellungen überführt. Bei Eingang der Waren wird der Lagerbestand aktualisiert. Dadurch wird die Verfügbarkeit der Materialien für die Bearbeitung hergestellt. Fertigungsaufträge werden freigegeben. Damit verbunden ist die Erstellung von Auftragspapieren für die Fertigung. Aufgrund der Auftragsfreigabe wird in einer Simulationsplanung unter Einbeziehung des aktuellen Belegungsplans, der Auftragspositionen und deren Zuordnungen zu den Maschinen und Bearbeitungsplätze ein Simulationsplan erzeugt, der anschließend in einen Maschinenbelegungsplan übernommen wird. Durch diese Übernahme entsteht ein Ressourcenbedarf. Unter Zuhilfenahme des Belegungsplans und der Zuordnungen von NC-Programmnummern zu den geplanten Auftragspositionen kann der Bedarf an NC-Programmen bestimmt werden. Ein solcher Bedarf kann auch durch die manuelle Auslösung einer NC-Bearbeitung mit einem bestimmten NC-Programm erzeugt werden. Ist das NC-Programm nicht im Leitstand vorhanden, so wird eine Programmanforderung ausgelöst, die zur Übertragung des Programms in den Leitstand führen muß. Das übertragene NC-Programm muß übernommen werden. Dabei werden bereits vorhandene Versionen gelöscht. Durch die Übernahme wird auch die Werkzeugeinsatzliste für das Leitsystem

verfügbar. Mit dieser und aus den Terminen aus dem Belegungsplan wird der Werkzeugbruttobedarf im Zeitablauf ermittelt. Dieser wird mit dem Werkzeugbestand in den Maschinen abgeglichen und führt zum Werkzeugnettobedarf. Wiederum unter Berücksichtigung der Werkzeugbestände in den Maschinen wird die Werkzeugein- und -auslagerungsliste für die Maschinen erstellt. Ausgehend von der Auslagerungsliste werden nach Fertigmeldung der vorhergehenden Arbeitsgänge die Werkzeuge ausgelagert und durch die Werkzeugdemontage zu Werkzeugbauteilen demontiert. Die Werkzeuge auf der Einlagerungsliste werden bereitgestellt. Dies umfaß die Montage aus Werkzeugbauteilen und die Vermessung unter Berücksichtigung der Werkzeugstammdaten. Die so erzeugten Werkzeug-Ist-Daten und Werkzeuge werden in die Maschinen übertragen bzw. eingelagert. Der Werkzeugbestand in den Maschinen wird aktualisiert und die Werkzeug-Ist-Daten befinden sich in der Maschinensteuerung. Nachdem auch der DIN-Code des NC-Programmes durch die DNC-Übertragung in die Maschinensteuerung übertragen wurde, die Auftragspapiere vorhanden sind und die Materialverfügbarkeit sichergestellt ist, kann durch den Startbefehl der Arbeitsgang ausgeführt werden. Nach der Ausführung werden Rückmeldungen erzeugt, die zu einer Prüfung auf Abweichungen vom Plan auf Auftragsebene führt. Eventuelle Planabweichungen führen zu einer Anpassung der aktuellen Kapazitätsauslastug und zu einem Planungsbedarf für PRP und den Maschinenbelegungsplan.

In diesen Ablauf der Auftragsbearbeitung fügt sich das Schnittstellensystem ein. Die Eingangs- und Ausgangsstellen des Schnittstellensystems sind in dem Netz-Ausschnitt in Anhang erkennbar. Die Transition „Systeme koppeln“ ist in einem weiteren Petri-Netz verfeinert (Anhang). Dieses Netz stellt den Prozeß der Systemkopplung dar. Ab dieser Verfeinerungsstufe sind für die Objektspeicher die Datenstrukturen angegeben. Die Zuordnung der Daten zu den Objektspeichern ist im Anhang aufgeführt. Objektspeichern, denen keine Entities zugeordnet werden können, lassen sich dennoch strukturieren. INCOME stellt hierzu unabhängige Datenelemente zur Verfügung, die einem Objektspeicher zugeordnet werden können. Diese Datenelemente und ihre Zuordnungen zu den Objektspeichern sind im Anhang aufgeführt.

Der Austauschprozessor für das PPS-System erzeugt bei Auftragsfreigabe aus den Arbeitsplänen, den Arbeitsplanpositionen, deren Zuordnungen zu NC-Programmen und Bearbeitungsplätzen sowie den Auftragsterminen eine lineare Auftragsübergabedatei. Alternativ kann eine Arbeitsplanübergabedatei aus den Arbeitsplänen, den Arbeitsplanpositionen und deren Zuordnungen zu NC-Programmen und Bearbeitungsplätzen erzeugt werden. Diese Übergabedateien werden durch das Schnittstellensystem gelesen und die gelesenen Fertigungsaufträge, Arbeitspläne, Aufträge, und Zuordnungen zu NC-Programmen und Bearbeitungsplätzen gespeichert.

Beim Einlesen eines Fertigungsauftrags wird zusätzlich der Status und die geplanten Termine jeder Auftragsposition im Schnittstellensystem geführt. Für das Einlesen der NC-Programmzuordnungen ist die Kenntnis des vorhandenen NC-DIN-Codes im Schnittstellensystem notwendig. Weiterhin wird beim Einlesen eines Fertigungsauftrags eine Auftragsfreigabeaufforderung erzeugt. Diese kann auch durch eine nachträgliche manuelle Freigabe des Auftrags erzeugt werden. Aufgrund einer solchen Auftragsfreigabeaufforderung wird mittels der gespeicherten Aufträge, Arbeitspläne, Arbeitsplanpositionen und deren Zuordnungen die Verfügbarkeit der NC-Programme im Schnittstellensystem geprüft. Sind die erforderlichen Daten verfügbar, so wird eine lineare Übergabedatei mit den genannten Daten erzeugt. Diese wird durch das Leitstandsystem gelesen. Die Daten werden dann im Leitstand gespeichert. Das Erzeugen der Übergabedatei erzeugt gleichzeitig einen neuen Status des Fertigungsauftrags im Schnittstellensystem.

NC-Programme gelangen auf zwei verschiedene Arten in die Datenbank des Schnittstellensystems, sowohl das NC-Programmiersystem als auch die Leitstände übertragen NC-Programme die gelesen werden. Beide Systeme erzeugen über Austauschprozessoren eine lineare Übergabedatei die eingelesen wird. Eventuelle Differenzen mit vorhandenen Versionen werden dem NC-Programmierer zur Verfügung gestellt und die gelesenen Daten im Schnittstellensystem gespeichert. Der Benutzer kann für ein bestimmtes Programm eine neue aktuelle Version vorgeben, die im Schnittstellensystem gesetzt werden muß. Erzeugt das Leitstandsystem eine NC-Programmanforderung, so erzeugt das Schnittstellensystem eine NC-Übergabedatei, die durch das Leitstandsystem gelesen wird. Damit steht das NC-Programm dann im Fertigungsleitstand zur Verfügung.

Bei Abarbeitung einer Auftragsposition erzeugt der Fertigungsleitstand Rückmeldungen. Aus diesen wird eine Übergabedatei erzeugt, die wiederum vom Schnittstellensystem gelesen wird. Dies führt zu einem neuen Status für eine Auftragsposition im Schnittstellensystem und zu einem Rückmeldebedarf an das PPS-System, falls sich die erzeugte Rückmeldung auf den ganzen Auftrag bezieht. Die gespeicherten Statusdaten werden daraufhin in eine Rückmeldedatei geschrieben, die duch das PPS-System gelesen werden muß, damit die Rückmeldungen auf Auftragsebene im PPS-System zur Verfügung stehen.

Wichtige Anforderung an das Schnittstellensystem war die Möglichkeit der Zuordnung von NC-Programmen zu Arbeitsplanpositionen. Dies geschieht unter Berücksichtigung des aktuellen Status und der Bearbeitungsplatzzuordnung und aktualisiert die gespeicherten Zuordnungen.

Der Prozeßablauf der Systemkopplung ist wiederum verfeinert. Verfeinerte Transitionen sind graphisch erkennbar. Die Transition „Auftragsübergabedatei lesen“ ist durch das in Anhang dargestellte Petri-Netz verfeinert. Ausgangsdatum ist die lineare Übergabedatei. Aus dieser wird die Nummer des Auftrages gelesen und anhand dieser Nummer wird in den im Schnittstellensystem gespeicherten Aufträgen geprüft, ob die Nummer bereits vergeben ist. In diesem Fall wird die Datei abgewiesen und als abgewiesene Datei gekennzeichnet. Ist die Auftragsnummer noch nicht vergeben, so wird die Arbeitsplannummer aus der Übergabedatei gelesen und mittels der im Schnittstellensystem gespeicherten Arbeitspläne wird geprüft, ob die Arbeitsplannummer bereits vergeben ist. Ist bereits ein Arbeitsplan mit der übergebenen Nummer vorhanden, wird geprüft, ob dazu bereits ein Backup-Arbeitsplan exitiert. Ist dieser vorhanden, wird er zusammen mit den zugehörigen Arbeitsplanpositionen gelöscht. Damit fehlt nun der Backup-Arbeitsplan. Ist kein Backup-Arbeitsplan vorhanden, wird der aktuell gespeicherte Arbeitsplan mit den zugehörigen Arbeitsplanpositionen als Backup-Arbeitsplan gekennzeichnet und somit der aktuell gespeicherte Arbeitsplan gelöscht. Wenn kein Arbeitsplan mit der angegeben Nummer vorhanden ist, werden die restlichen Daten aus der Übergabedatei gelesen. Diese beinhalten den Auftrag, den zugeordneten Arbeitsplan mit Arbeitsplanpositionen, die Bearbeitungsplatz- und NC-Programmzuordnungen dieser Positionen. Des weiteren wird der Status jeder Position gesetzt und die Plantermine gelesen. Für das Lesen der Daten werden die vorhandenen NC-Programme benötigt, bzw. es werden neue Versionen erzeugt, wie in der Verfeinerung der Transition „Auftragsdaten lesen“ deutlich werden wird. Nach dem Lesen der Daten soll der Auftrag freigegeben werden. Dieser Objektspeicher wird im übergeordneten Diagramm weiterverwendet. Es entsteht ferner ein Datenabgleichsbedarf für die Arbeitsplanpositionszuordnungen mit den jeweiligen Vorgängerversionen. Dazu muß geprüft werden, ob Vorgänger, also solche mit Backup-Kennung, vorhanden sind. Ist dies nicht der Fall wird nach einem ähnlichen Arbeitsplan gesucht. Wenn entweder ein ähnlicher Arbeitsplan gefunden wurde oder eine Backup-Version vorhanden war, wird ein Befehl zur Übernahme der Zuordnungen auf die aktuellen Arbeitsplanpositionen erzeugt. Dies führt zu neuen Bearbeitungsplatz- und NC-Programmzuordnungen.

Die verfeinerte Transition „Auftragsdaten lesen“ ist durch das Netz in Anhang dargestellt. Wenn der Arbeitsplan mit der übergebenen Nummer noch nicht im Schnittstellensystem vorhanden ist, werden aus der Übergabedatei die Zuordnungen der Arbeitsplanpositionen zu den NC-Programmen gelesen. Nun wird geprüft, ob das angegebene NC-Programm bereits im Schnittstellensystem vorliegt. Ist dies nicht der Fall wird die Programmnummer und eine Version 0 des Programms in die Datenbank des Schnittstellensystems eingefügt. Damit ist das NC-Programm vorhanden und der

Rest der Daten kann aus der Übergabedatei gelesen werden. Diese beinhalten den Auftrag, den zugeordneten Arbeitsplan mit Arbeitsplanpositionen, die Bearbeitungsplatz- und NC-Programmzuordnungen dieser Positionen. Die Auftragsfreigabeaufforderung und der Datenabgleichsbedarf werden auf übergeordneter Netzebene weiterverwendet.

Die Vorgehensweise für das Lesen von Arbeitsplänen welche ohne zugehörigen Auftrag übergeben wird, ist analog zu der für Aufträge in Verbindung mit Arbeitsplänen und ist durch die Petri-Netze in den Anhängen und dargestellt.

Die Prozesse des Einlesens und Übertragens von NC-Dateien sind durch Verfeinerungen des Netzes „Systeme koppeln“ in den Anhängen und dargestellt. Die Übergabedatei wird auf Vollständigkeit geprüft. Dies beinhaltet die Prüfung, ob alle sechs zusammengehörigen Teile vorhanden sind. Ist die Übergabedatei nicht vollständig, wird sie abgewiesen und als solche gekennzeichnet. Ist die Übergabedatei vollständig, so wird die Programmnummer aus der Datei gelesen. Anhand der vorhandenen NC-Programme wird ermittelt, ob diese Programmnummer bereits im Schnittstellensystem vorhanden ist. Ist sie es nicht, ist dieses die erste Version und erhält die Versionsnummer 1,0. Falls eine Version vorhanden ist, wird deren Versionsnummer ermittelt. Ist nur die Version 0 vorhanden, wird diese gelöscht. Damit ist keine Version mehr vorhanden. Ist eine andere Version als die 0 vorhanden, wird anhand der Übergabedatei das Herkunftssystem festgestellt. Ist das Herkunftssystem das NC-Programmiersystem, dann ist die neue Versionsnummer die um 1 erhöhte Hauptversionsnummer der letzten im Schnittstellensystem gespeicherten Version. Ist das Herkunftssystem ein Leitstand, dann ist die neue Versionsnummer die in der Nebenversionsnummer um 1 erhöhte Versionsnummer der letzten im Schnittstellensystem gespeicherten Version. Nachdem die neue Versionsnummer ermittelt worden ist, werden die Daten aus der Übergabedatei gelesen. Der DIN-Code wird im Schnittstellensystem gespeichert. Gleichzeitig entsteht ein Abgleichbedarf mit der Vorgängerversion um evtl. Differenzen zu ermitteln.

Das Schreiben von NC-Programmen in Übergabedateien für die Leitstandsysteme erfolgt bei Vorliegen einer Anforderung durch den Leitstand. Die aktuelle Version des Programmes wird anhand der gespeicherten Versionen ermittelt. Danach werden die zu dieser Version gehörigen Daten in die Übergabedatei geschrieben.

Die Behandlung der fertigungsunterstützenden Daten (FUD), derzeit Texte und Grafiken, wird nicht gesondert modelliert, denn sie erfolgt analog zur Behandlung der NC-Programme.

Organisationssicht

Die Organisationssicht ist durch das Organigramm in Anhang dargestellt. Eine integrierte Darstellung wird erreicht indem den Transitionen Organisationseinheiten zugeordnet werde, welche die Funktionen oder Aktivitäten ausführen. Den Organisationseinheiten wiederum sind die eingesetzten EDV-Systeme als Ressourcen zugeordnet. Damit wird dargestellt, zwischen welchen Bereichen der Auftragsbearbeitung ein Systemwechsel besteht, der durch das Schnittstellensystem zu überbrücken ist[142].

Es wird deutlich, daß die Steuerungssicht, wie im ARIS-Modell und in den GoM gefordert[143], die verschiedenen Sichtweisen integriert. Neben der Integration der Organisationssicht wird auch die Datensicht integriert.

DV-Konzept und Implementierung

Datensicht

Um eine redundanzfreie und konsistente Datenverwaltung zu ermöglichen, wird die Speicherung mittels eines relationalen Datenbankmanagementsystems (DBMS) und einer relationalen Datenbank realisiert. Dies ermöglicht insbesondere eine einfache Überprüfung und Sicherstellung der Datenintegrität, denn das Relationenmodell verfügt bereits über Mechanismen, die Daten auf die in angeführten Integritätsbedingungen zu prüfen und deren Einhaltung sicherzustellen. Konkret wird das System Oracle in der Version 7 eingesetzt, da dieses unternehmensweit favorisiert wird und bereits in verschiedenen anderen Systemen im Einsatz ist, so daß das notwendige Know-How im Unternehmen zur Verfügung steht.

Relationenmodell[144]

Das relationale Modell geht auf Arbeiten von E.F. Codd zurück und bedient sich Elementen der Mengenalgebra, eben den Relationen. Eine Relation ist definiert als eine Menge von Tupeln über einer Attributmenge, dem Relationenschema. Ein Relationenschema entspricht somit einem Entity-Set im ER-Modell. Datenabhängigkeiten können in intrarelationale und interrelationale Abhängigkeiten unterschieden werden. Intrarelationale Abhängigkeiten sind insbesondere Schlüsselabhängigkeiten und Wertebereichrestriktionen. Als ein Schlüssel eines

142 Vgl. Anhang .
143 Vgl. Kapitel und Kapitel .
144 Vgl. Vossen (1994), S. 134ff.

Relationenschemas wird eine Menge von Attributen bezeichnet, durch die ein Tupel einer Relation eindeutig identifizierbar ist. Gibt es mehrere solcher Mengen von Attributen, wird eine von ihnen als Primärschlüssel gekennzeichnet. So findet der Schlüsselbegriff im ER-Modell sein Analogon im relationalen Modell. Interrelationale Abhängigkeiten dienen zur Sicherung der referentiellen Integrität und sind u.a. Inklusionsabhängigkeiten, insb. Fremdschlüsselbeziehungen. Eine Menge von Attributen eines Relationenschemas wird als Fremdschlüssel bezeichnet, wenn diese Menge gleichzeitig Schlüssel eines weiteren Relationenschemas ist.

Um aus dem in Kapitel erstellten Entity-Relationship-Modell ein Relationenmodell zu erstellen, bedient man sich Transformationsregeln, die hier kurz angegeben werden sollen:[145]

- Jedes Entity-Set wird in ein Relationenschema transformiert. Die identifizierenden Attribute bilden zusammen den Primärschlüssel dieses Schemas.

- Jede Beziehung wird in ein Relationenschema transformiert dessen Attribute aus den identifizierenden Attribute der beiden beteiligten Entities gebildet werden. Diese Vorgehensweise ist für n:m-Beziehungen notwendig, für 1:n-Beziehungen läßt sich jedoch eine Vereinfachung angeben. Dabei wird das Relationenschema desjenigen Entity-Typs, welches einfach in die Beziehung eingeht, um die Attribute erweitert, die aus den identifizierenden Attributen des anderen Entity-Sets gebildet werden (Fremdschlüsselbeziehung).

Das mittels dieser Regeln transformierte ER-Modell ist als Relationendiagramm mit Fremdschlüsselbeziehungen im Anhang graphisch dargestellt. Aus den Relationenschemata werden die Attribute deutlich, die als Fremdschlüssel aus anderen Relationenschemata eingehen. Sie enthalten als Präfix den Namen des Relationenschemas in dem sie ursprünglich definiert sind. So lassen sich Fremdschlüsselbeziehungen leicht verfolgen. Bei den Relationenschemata ist im Kopf jeweils die Datenbank angegeben, in der die Relationen implementiert werden sollen, hier „INTEGDB“. Eine textuelle Beschreibung mit den inter- und intrarelationalen Abhängigkeiten, insb. den Wertebereichen der Attribute und den Schlüsseln und Schlüsselbeziehungen ist in Anhang wiedergegeben.

Integritätsüberwachung

Neben der Integritätssicherung im Datenbankschema besteht eine weitere Möglichkeit darin, Integritätsbedingungen bei jeder Operation auf der Datenbank zu überprüfen und

[145] Vgl. ausführlich Vossen (1994), S. 134ff.

die auszuführende Operation ggf. zurückzuweisen, falls durch deren Auswirkungen die Bedingungen verletzt werden. Diese Integritätsüberwachung läßt sich durch das Triggerkonzept realisieren. Ein Trigger ist ein in der Datenbank hinterlegtes Modul dessen Ausführung durch bestimmte Datenbankoperationen ausgelöst wird.

Datenbanktrigger lassen sich sowohl in die Steuerungssicht als auch in die Funktionssicht einordnen. Einerseits legt ein Trigger fest, wann ein Modul auszuführen ist. Andererseits ist in einem Modul der Funktionssicht die auszuführende Funktionalität beschrieben. Die für das Schnittstellensystem definierten Module sind im Anhang , die erstellten Trigger sind in Anhang aufgeführt. Die verwendeten Trigger erzeugen bei Verletzung der überprüften Integritätsbedingungen einen Fehler und machen damit die Effekte der sie auslösenden Befehle rückgängig.

Der Trigger AVGPLATZTRIG, der die Ausführung des Moduls AVGTYP-PLATZZUORDNUNG PRUEFEN auslöst, wird nach dem Löschen eines Datensatzes aus der Tabelle ZAVGTYPEN_BEARBEITUNGSPLAETZE ausgelöst. Er stellt sicher, daß die Zuordnungen von einem Bearbeitungsplatz zu einer darauf ausführbaren Arbeitsvorgangsart nicht gelöst wird, solange noch Arbeitsplanpositionen auf diesen Bearbeitungsplatz verweisen.

Der Trigger PLATZAVO, der die Ausführung des Moduls PLATZ-AVO-ZUORD PRUEFEN auslöst, wird nach jeder Einfüge- bzw. Änderungsoperation auf der Tabelle ZAVGTYPEN_BEARBEITUNGSPLAETZE ausgelöst. Er stellt sicher, daß die Menge der möglichen Bearbeitungsplätze, die zu jeder Arbeitsplanposition angegeben sind, nur eine einzige Fertigungsinsel referenzieren. Weiterhin wird überprüft, daß die über die Zuordnungen von Arbeitsplanpositionen und Arbeitsvorgangsarten sowie von Arbeitsplanpositionen mit möglichen Bearbeitungsplätzen und dann mit Arbeitsvorgangsarten gebildeten Mengen von möglichen Arbeitsvorgangsarten übereinstimmen.

Der Trigger BEAPLATZ_TRIGGER, der die Ausführung des Moduls BEARBEITUNGPLAETZE PRUEFEN auslöst, wird nach jeder Operation auf der Tabelle ZBEARBEITUNGSPLAETZE ausgelöst. Er stellt sicher, daß die Bezeichnungen in den Systemen Soflex und Triton für die Bearbeitungsplätze eindeutig sind.

Der Trigger BEATRIGGER, der die Ausführung des Moduls BEAVER TABELLE PRUEFEN auslöst, wird nach jeder Operation auf der Tabelle ZBEAVER ausgelöst. Er stellt sicher, daß zu jeder FUD genau eine aktuelle Version existiert.

Der Trigger NCTRIGGER, der die Ausführung des Moduls NCNCVER PRUEFEN auslöst, wird nach dem Löschen eines Datensatzes aus der Tabelle ZNCNRVER ausgelöst. Er stellt sicher, daß zu jeder NC-Programmnummer genau eine aktuelle Version existiert.

Datenbankimplementierung

Die Implementierung besteht aus dem physischen Entwurf und dem sog. Einrichten der Datenbank, d.h. der Erzeugung der Datenstrukturen mittels der Datendefinitionssprache (DDL) des verwendeten DBMS. Mit dem CASE-Werkzeug werden aus den Angaben der Schlüsselattribute der Relationenschemata die für einen schnellen Zugriff notwendigen Indexstrukturen erzeugt. Die Relationenschemata werden in der Datenbank durch Tabellen abgebildet. Diese enthalten im konkreten Fall das Präfix Z, welches für das schnelle Auffinden in alphabetisch sortierten Listen nützlich ist[146]. Die Triggerbeschreibung mittels DDL wird aus den Triggerdefinitionen und den in den Modulen hinterlegten SQL-Befehlen erzeugt. Neben den durch die ANSI genormten SQL-Befehlen werden dabei auch die Oracle-eigenen prozeduralen Ergänzungen zu SQL, PL/SQL (procedural logic/structured query language) verwendet. Die erzeugten SQL-Befehle sind den Listings des Anhangs zu entnehmen.

Steuerungssicht

Wegen der in genannten Restriktionen durch die eingesetzten Werkzeuge, insb. in den Bereichen der Dateibehandlung und der Gestaltung von Benutzeroberflächen, wurde die gesamte Ablauflogik manuell in der Programmiersprache Visual Basic der Firma Microsoft implementiert und ist ausschnittsweise in den Anhängen und dargestellt.[147]

Das Schnittstellensystem ist in fünf Module geteilt, die jeweils als selbständige Programme implementiert sind. Neben dem Administrationsmodul sind in der Grafik auch die vier Koppelprozessoren erkennbar.

---

146 Die war insb. in der Entwicklungszeit hilfreich, kann aber auch für spätere Wartungsarbeiten sinnvoll sein.

147 Aus Platzgründen kann hier nur ein Teil des erstellten Codes wiedergegeben werden. Der ABB Kraftwerke AG liegt der gesamte Quellcode vor.

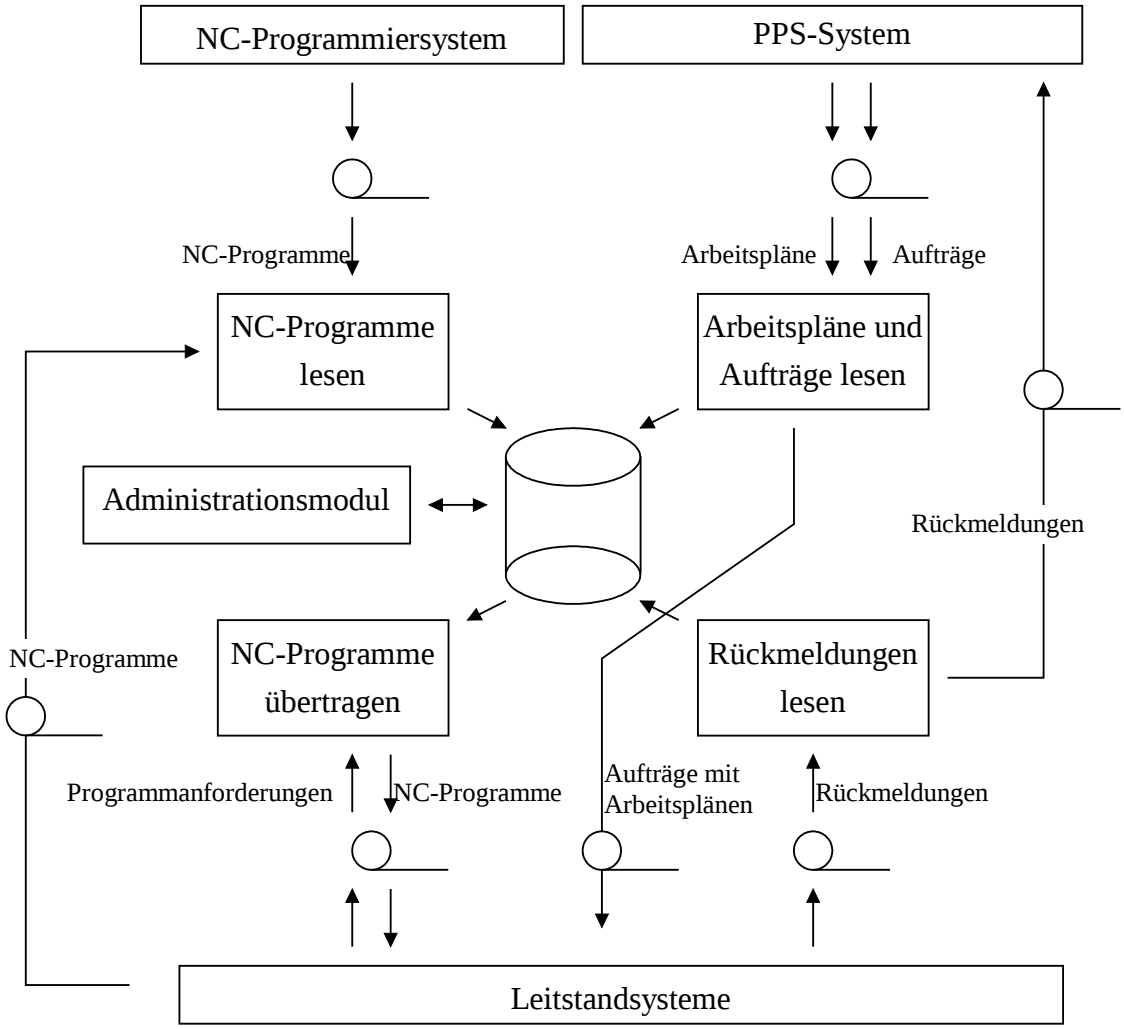


Abb. 3.1: Architektur des Schnittstellensystems

Das Kernstück ist das Administrationsmodul, welches die Benutzerschnittstelle zu den Stammdaten des Schnittstellensystems und die Sicht auf die Arbeitsdaten im System darstellt. In diesem Modul lassen sich die Stammdaten verwalten, also insbesondere die Fertigungsinseln mit allen Attributen anlegen, die Bearbeitungsplätze mit den jeweiligen Namen anlegen, die Bearbeitungsstationen und die Arbeitsvorgangstypen anlegen. Weiter können die Zuordnungen zwischen diesen Daten hergestellt werden, z.B. die Zuordnung der Bearbeitungsstationen zu den Arbeitsvorgangstypen oder die Bearbeitungsplätze zu den Stationen und den Fertigungsinseln. Zu den Stammdaten zählen weiter die zugelassenen Daten- und Dateitypen für die fertigungsunterstützenden Daten (FUD), derzeit Texte und Grafiken. Auch die Arten der zulässigen NC-Programminformationen lassen sich hier eintragen, derzeit NC-DIN-Code, Werkzeugeinsatzliste, Vorrichtungseinsatzliste etc. Diese Daten müssen bei Inbetriebnahme des Schnittstellensystems oder bei Änderungsbedarf eingegeben bzw. angepaßt werden. Im Anhang sind beispielhaft einige der für den Benutzer sichtbaren

Bildschirmmasken dargestellt. Des weiteren lassen sich in diesem Modul auch die Zuordnungen der Arbeitsplanpositionen zu NC-Programmdateien und den FUD erstellen und lösen. Es sind Funktionen für das Sperren und Entsperren von FUD und für das Setzen der jeweils aktuellen Version von NC-Programmversionen und FUD vorhanden. Für diese Daten existiert eine Funktion zum Übertragen der Daten auf einen beliebigen Datenträger. Damit wird insbesondere der manuelle DNC-Betrieb mit den Fertigungsleitständen ermöglicht.

Ein weiteres Modul übernimmt das Einlesen von NC-Programmen. Dies sind sowohl die Programme, die vom NC-Programmiersystem übergeben werden, wie auch solche, die die Leitstandsysteme zurückübertragen. Dazu wird in in einer Endlosschleife ein Dateiverzeichnis auf einem Dateiserver in regelmäßigen zeitlichen Abständen auf das Vorhandensein von Dateien untersucht und versucht, diese zu lesen. Findet das Modul Fehler in den Dateien, werden sie in ein Fehlerverzeichnis kopiert und es wird eine Fehlermeldung in eine Protokolldatei geschrieben. Arbeitsplandaten und Fertigungsauftragsdaten werden durch ein weiteres Programm auf ähnliche Art und Weise eingelesen. Alle wichtigen Parameter der Module, z.B. das zeitliche Intervall, der Name der Protokolldatei, die Verzeichnisnamen, sind über Parameterdateien individuell einstellbar, so daß sich die Module ohne großen Aufwand in andere Umgebungen übertragen lassen.

Die beiden Programme für die Bearbeitung von Anforderungen für fertigungsunterstützende Daten und NC-Programme sowie für die Bearbeitung von Rückmeldungen befinden sich derzeit in der Implementierungsphase, werden aber nach dem gleichen Prinzip arbeiten. Alle vier Koppelprozessoren werden auf einem dedizierten Rechner unter einem Multitaskingbetriebssystem[148] gleichzeitig ablaufen, um so optimale Aktualität der Daten gewährleisten zu können.

[148] Derzeit ist Windows/NT der Firma Microsoft vorgesehen.

Ergebnis und Ausblick

Aufgrund der Notwendigkeit, die bestehenden Produktionsplanungssysteme in einem sinnvollen Produktionsplanungsprozeß zu integrieren, wurde ein Schnittstellensystem entwickelt, welches diese Integration herstellt. Durch genaue Analyse der bestehenden Systeme wurde Handlungsbedarf identifiziert. Mit den Methoden des Software-Engineering wurde ein evolutionärer Prototyp entwickelt, der diesen Handlungsbedarf deckt. In einer Phase der Anforderungsanalyse wurden die fachlichen Anforderungen an das Schnittstellensystem formuliert. Diese wurden in einem Fachkonzept formalisiert und dann in ein DV-Konzept umgesetzt und anschließend implementiert. Der enstandene Prototyp kann die heutigen, kurzfristigen Anforderungen an die informationstechnische Verbindung der PPS-Komponenten abdecken. Er ist aber auch für die zukünftigen mittelfristigen Anforderungen gerüstet.

Neben dem fertiggestellten Prototypen hat insb. die intensive Bestandsaufname und explorative Erforschung der bestehenden Systeme im Rahmen dieser Arbeit einen großen Beitrag zur Feststellung der genauen Fähigkeiten und dem Verständnis der bestehenden Systeme geleistet. Durch die notwendige Integration des Schnittstellensystems in den Planungsprozeß wurde die Dringlichkeit der Festlegung dieses Prozeßes unter Beachtung der Fähigkeiten der existierenden Systeme deutlich. Ausgehend von den gewonnen Erkenntnissen über die Möglichkeiten der Planung mit den vorhandenen Systemen wurde in Zusammenarbeit mit den Fachabteilungen der grobe Ablauf der Produktionsplanung im Unternehmen festgelegt und vorläufig verabschiedet. Dadurch können nun verstärkt die EDV-Systeme zur Unterstützung des derzeit zu großen Teilen manuell durchgeführten Planungsablaufs herangezogen werden.

Der Prototyp befindet sich derzeit in der letzten Phase des Systemtests und wird in Kürze der Fachabteilung zur Begutachtung und Evaluierung vorgestellt werden. Hieraus werden neue Rückschlüsse auf evtl. fehlende Funktionalitäten erwartet, die im Anschluß in das Fachkonzept aufzunehmen sind. Der Prototyp muß dann aufgrund dieses neuen Fachkonzepts evolutionär weiterentwickelt werden.

Einige Module, die weitgehend unabhängige Teilfunktionen des Schnittstellensystems realisieren, insb. die Koppelprozessoren für das Lesen und Übertragen der Rückmeldungen sowie für das Beantworten der Anforderungen von fertigungsunterstützenden Daten, befinden sich noch in der Implementierungsphase. Kurzfristig werden die fertigungsunterstützenden Daten, derzeit Texte und Grafiken, bei der Übertragung von Aufträgen mit diesen zusammen an das Fertigungsleitsystem übertragen, ungeachtetet dessen, ob sie dort bereits vorliegen. Dies geschieht nicht

zuletzt deshalb, weil die Leitstandsysteme die in deren Pflichtenheft geforderte Funktionalität zur Datenanforderung noch nicht besitzen und auch noch nicht in der Lage sind, Rückmeldedateien zu erzeugen. Der Hersteller der Fertigungsleitstände wird diese Funktionalität jedoch nachliefern. Die Datenübertragung geschieht bis dahin unter der Gefahr, daß die Speicher- und Verwaltungskapazitäten der FLS dadurch stark belastet werden, garantiert aber die Verfügbarkeit der Daten.

Die Implementierung der Dateischnittstelle des Triton-Systems aufgrund der entworfenen Schnittstellenspezifikation ist derzeit in der Vorbereitung. Ausgehend von den benötigten Daten werden in einem DV-Konzept die betroffenen Tabellen der Triton-Datenbank identifiziert und eine Implementierungsstrategie entwickelt. Gleichzeitig wird an einer Fertigungshilfsmittelverwaltung im Triton-System gearbeitet. Diese wird die Fertigungshilfsmittel zwar nicht dispositiv verwalten können, wird aber die Angabe der Beistellmengen zu einer Arbeitsplanposition ermöglichen.

Literaturverzeichnis

Abeln, O.: Die CA...-Techniken in der industriellen Praxis: Handbuch der computergestützten Ingenieur-Methoden. München u.a. 1990.

Adam, D. (PRODUKTION): Produktionsmanagement. 7. Aufl., Wiesbaden 1993.

Adam, D. (PLANUNG): Planung und Entscheidung. Modelle-Ziele-Methoden. 3. Aufl., Wiesbaden 1993.

Adam, D. (FFS): Flexible Fertigungssysteme (FFS) im Spannungsfeld zwischen Rationalisierung, Flexibilisierung und veränderten Fertigungsstrukturen. In: SzU. Hrsg.: D. Adam. Wiesbaden 1993. (Band 46).

Arning, A.: Die wirtschaftliche Bewertung der Zentrenfertigung: dargestellt am Beispiel einer Fertigungsinsel. In: Bochumer Beiträge zur Unternehmensführung und Unternehmensforschung. Wiesbaden, 1987. (Band 30).

Balgheim, N.; Jansen, F.J.: Integrierter Auftrags-Informationsfluß mit PPS-, FLS- und BDE-Systemen. CIM Management 9 (1993) 1, S. 11-15.

Barker, R.: CASE*Method. Entity Relationship Modellierung. Bonn u.a. 1992.

Barker, R.: CASE*Method. Tasks and deliverables. In: Computer Aided Systems Engineering. Hrsg.: Oracle Corporation. Wokingham u.a. 1990.

Becker, J.: CIM-Integrationsmodell - Die EDV-gestützte Verbindung betrieblicher Bereiche. Berlin u. a. 1991.

Becker, J.; Priemer, J.: Die universelle CIM-Schnittstelle - mehr als ein Data Dictionary? HMD. 161 (1991). S. 144-155.

Becker, J.; Vossen, G.: Geschäftsprozeßmodellierung und Workflow-Management: Eine Einführung. In: Geschäftsprozeßmodellierung und Workflow-Managment. Hrsg.: G. Vossen, J. Becker. Bonn u.a. 1996.

Becker, J.; Rehfeld, M.; Turowski, K.: Koordination verteilter Objekte in der PPS. CIM Management. 11 (1995) 3. S. 33-37.

Becker, J.; Rosemann, M. (CIM): Logistik und CIM. Die effiziente Material- und Informationsflußgestaltung im Industrieunternehmen. Berlin u. a. 1993.

Becker, J.; Rosemann, M. (FFS): Organisatorische Integration von Flexiblen Fertigungssystemen durch CIM und Logistik. In: SzU. Hrsg.: D. Adam. Wiesbaden 1993. (Band 46).

Becker, J.; Schütte R.: Handelsinformationssysteme. Landsberg/ Lech 1996.

Berztiss, A.: Software methods for business reengineering. New York u.a. 1996.

Ferstl, O.K.; Mannmeusel, T.: Dezentrale Produktionslenkung. CIM Management. 11 (1995) 3, S. 26-32.

Förster, H.-U.: Integration von flexiblen Fertigungszellen in die PPS. In: Forschung für die Praxis. Hrsg: R. Hackstein. Berlin u.a., 1988. (Band 19).

Kassel, S.: Multiagentensysteme als Ansatz zur Produktionsplanung und -steuerung. Information Management. o.Jg. (1996) 1, S. 46-50.

Keller, G.; Kern, S.: Dezentrale Inselstrukturen in Planung und Fertigung. In: Fertigungssteuerung, Expertenwissen für die Praxis. Hrsg.: A.-W. Scheer. München, Wien 1991.

Kurbel, K.: Produktionsplanung und -steuerung - Methodische Grundlagen von PPS-Systemen und Erweiterungen. In: Handbuch der Informatik. Hrsg: A. Endres, H. Krallmann, P. Schnupp. München, Wien 1993 (Band 13.2).

Martin, J.A.: Information Engineering: A Trilogy. Band 1. Englewood Cliffs, NJ, 1989.

Maßberg, W (Hrsg.): Fertigungsinseln in CIM-Strukturen. In: CIM-Fachmann. Hrsg.: I. Bey. Berlin u.a., 1993.

Milberg, J. (Hrsg.): Von CAD/CAM zu CIM, In: CIM-Fachmann. Hrsg.: I. Bey. Berlin u.a., 1992.

Moser, M.: Entscheidungsunterstützung für die dezentrale Fertigungssteuerung. CIM Management. 11 (1995) 5, S. 69-73.

o.V.: ABB Jahresbericht. Hrsg.: ABB AG. Mannheim 1993.

o.V.: Innovativ für Kraft und Wärme. Hrsg.: ABB Kraftwerke AG. Mannheim 1995.

Reisig, W.: Systementwurf mit Netzen. Berlin u.a. 1985.

Rosenstengel, B.; Winand, U.: Petri-Netze - Eine anwendungsorientierte Einführung, Braunschweig, Wiesbaden 1982.

Ruffing, T.: Die integrierte Auftragsabwicklung bei Fertigungsinseln - Grobplanung, Feinplanung, Überwachung. In: Fertigungssteuerung, Expertenwissen für die Praxis. Hrsg.: A.-W. Scheer. München, Wien 1991.

Ruland, D.: Entwicklung von CIM-Systemen mit Datenbankeinsatz: Grundlagen, Konzepte, Realisierung. München u.a. 1991.

Scheer, A.-W.: CIM, Der computergesteuerte Industriebetrieb. 4. Auflage, Berlin u.a. 1990.

Scheer, A.-W.: Wirtschaftsinformatik. Referenzmodelle für industrielle Geschäftsprozesse. 5. Aufl., Berlin u.a. 1994.

Scheer, A.-W.: Neue Architekturen für PPS-Systeme. In: Fertigungssteuerung, Expertenwissen für die Praxis. Hrsg.: A.-W. Scheer. München, Wien 1991.

Schneider, B.; Rinschede, M.: Kooperierende dezentrale Leitstände für ein flexibles Scheduling. Information Management. o.Jg. (1996) 5, S. 52-61.

Schönthaler, F.; Nemeth, T.:Software-Entwicklungswerkzeuge: Methodische Grundlagen. In: Leitfäden der angewandten Informatik. Hrsg: H.-J. Appelrath, L. Richter, W. Stucky. Stuttgart 1990.

Vossen, G.: Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme, 2. Aufl., Bonn u.a. 1994.

Westkämper, E.; Wiedenmann, H.: Dezentrale Organisation und ihre informationstechnische Unterstützung in der Produktionsplanung und -steuerung. Industrie Management. 12 (1996) 3. S. 39-42.

Anhang

Organigramm der Deutschen Asea Brown Boveri AG

Entity Relationship Diagramm des Schnittstellensystems

Entities und deren Attribute

Prozeßbeschreibung „Planungsablauf“

Prozeßbeschreibung „Systeme koppeln“

Prozeßverfeinerung „Systeme koppeln“

Prozeßbeschreibung „Auftragsübergabedatei lesen“

Prozeßbeschreibung „Auftragsdaten lesen“

Prozeßbeschreibung „Arbeitsplanübergabedatei lesen“

Prozeßbeschreibung „Arbeitsplandaten lesen“

Prozeßbeschreibung „NC-DIN-Code lesen“

Prozeßbeschreibung „NC-DIN-Code schreiben“

Datenstrukturen von Objektspeichern

```
Projekt:                    INTEGRATION (1)
Applikation:
Netz:                       %
Objektspeicher:             %
Bericht:                    Datenstrukturen von
                           Objektspeichern
Berichtsdatum:              28-JAN-97
Ausgearbeitet von:          INCOMEDBA

Speicher:          AVO-NC-Zuordnungen in IS

Datenstrukturen    Entity                      Atttribut
                   APL                         ARBEITSPLA
                                               BAK
                                               HERST_ID
                   AVO                         APL_INDEX
                   NCNR                        NCNR

Speicher:          Arbeitspläne in IS

Datenstrukturen    Entity                      Attribut
                   APL                         ARBEITSPLA
                                               BAK
                                               ERSTELLDZ
                                               ERSTELLER
                                               GESPERRT
                                               HERST_ID
                                               KOMMENTAR

Speicher:          Auftrag/Arbeitsplanpos. in IS

Datenstrukturen    Entity                      Attribut
                   APL                         ARBEITSPLA
                                               BAK
                                               HERST_ID
                   AVO                         APL_INDEX
                                               AUTOMATIK
                                               AVG_NR
                                               GESPERRT
                                               KOMMENTAR
                                               MESSFREQ
                                               STUECK
                                               STUECKZEIT

Speicher:          Bearbeitungsplatzzuord. in IS

Datenstrukturen    Entity                      Attribut
                   APL                         ARBEITSPLA
                                               BAK
                                               HERST_ID
                   AVO                         APL_INDEX
                   BEARBEITUNGSPLATZ           PLATZNR

Speicher:          NC DIN Code in IS

Datenstrukturen    Entity                      Attribut
                   BEAART                      DTK
                                               EDITOR
                   BEADAT                      DATUM
                   NCNR                        NCNR
                   NCNRNVER                    AKTVER
                                               ERSTELLDZ
                                               ERSTELLER
                                               GESPERRT
                                               KOMMENTAR
                                               LAUFDR
                                               MODFREIG
                                               STUECK
                                               VER
                                               VERWORT
                   STATFREIG                   STARTFREIG
                                               TRANSFREIG
```

```
Speicher:           Aufträge in IS

Datenstrukturen    Entity                       Attribut
                   APL                          ARBEITSPLA
                   AUFTRAG                      ABLIEFERORT
                                                BEZEICHNG
                                                ERSTELLDZ
                                                ERSTELLER
                                                FEFREI_IST
                                                FERTFREIG
                                                FTGAUFTRAG
                                                FTGSTUFE
                                                HERST_ID
                                                KOMMENTAR
                                                LIEFERUNG
                                                MANDANT
                                                PLANFREIG
                                                PLFREI_IST
                                                PRIORITAET
                                                PRIO_IST
                                                PROJEKT
                                                STATUS
                                                STUECK
                                                STUECK_IST

Speicher:           Auftragspositionsstatus in IS

Datenstrukturen     Entity                       Attribut
                    APL                          ARBEITSPLA
                                                 BAK
                                                 HERST_ID
                    AUFTRAG                      FTGAUFTRAG
                    AVO                          APL_INDEX
                    VIB                          DATUM
                                                 STATUS

Speicher:           Auftragspositionstermine in IS

Datenstrukturen     Entity                       Attribut
                    APL                          ARBEITSPLA
                                                 BAK
                                                 HERST_ID
                    AUFTRAG                      FTGAUFTRAG
                    AVO                          APL_INDEX
                    PLANUNGSTERMIN               ANFANG
                                                 ENDE_FRUEH
```

Ressourcen und Aktivitäten

```
Projekt:                      INTEGRATION (1)
Netz:                         Planungsablauf
Aktivität:                    %
Bericht:                      Aktivitäten
Berichtsdatum:                28-JAN-97
Ausgearbeitet von:            INCOMEDBA
-------------------------------------------------------------
Ablaufnetz:           Planungsablauf

Aktivität             APL Prüfen auf Vollständigkeit
                 Ressourcen            Name
  Art             Anz.  Beteilig.
                                       PPS System
  Ausführend      1         100 %

Aktivität             Aktivitäten/Baugrp. ermitteln
                 Ressourcen            Name
  Art             Anz.  Beteilig.
                                       PPS System
  Ausführend      1         100 %

Aktivität             Anpassen der Aktivitäten
```

	Ressourcen		Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

Aktivität Arbeitsgang ausführen

	Ressourcen		Name
Art	Anz.	Beteilig.	
			Fertigungsleitstand
Ausführend	1	100 %	

Aktivität Arbeitsplan erstellen

	Ressourcen		Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

Aktivität Arbeitsplan kopieren

	Ressourcen		Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

Aktivität Arbeitsplan verwalten

	Ressourcen		Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

Aktivität Arbeitsplanposition verwalten

	Ressourcen		Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

Aktivität Auftrag ABB-intern vergeben

	Ressourcen		Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

Aktivität Auftrag ablehnen

	Ressourcen		Name
Art	Anz.	Beteilig.	
			Vertrieb KW/PY
Ausführend	1	100 %	

Aktivität Auftrag annehmen

	Ressourcen		Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

Aktivität Auslastungsplanung durchführen

	Ressourcen		Name
Art	Anz.	Beteilig.	
			Vertrieb KW/PY
Ausführend	1	100 %	

Aktivität Belegungsplan aktualisieren

	Ressourcen		Name
Art	Anz.	Beteilig.	
			Fertigungsleitstand
Ausführend	1	100 %	

Aktivität Bestellung auslösen

	Ressourcen		Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

Aktivität Bestätigen und Überführen

	Ressourcen		Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

Aktivität DNC Übertragung

Ressourcen			Name
Art	Anz.	Beteilig.	
			Fertigungsleitstand
Ausführend	1	100 %	

Aktivität Ein-, Auslagerungsliste erst.

Ressourcen			Name
Art	Anz.	Beteilig.	
			Fertigungsleitstand
Ausführend	1	100 %	

Aktivität Fertigungsaufträge freigeben

Ressourcen			Name
Art	Anz.	Beteilig.	
			Fertigungsleitstand
Ausführend	1	100 %	

Aktivität Grobkalkulation

Ressourcen			Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

Aktivität Kapazitätsabgleich

Ressourcen			Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

Aktivität Kapazitätsabgleich PRP

Ressourcen			Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

Aktivität Kapazitätsbelastung anpassen

Ressourcen			Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

Aktivität Make-or-Buy Entscheidung

Ressourcen			Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

Aktivität Manuelle Auslösung DNC

Ressourcen			Name
Art	Anz.	Beteilig.	
			Fertigungsleitstand
Ausführend	1	100 %	

Aktivität NC-Compilierung

Ressourcen			Name
Art	Anz.	Beteilig.	
			NC Programmiersystem
Ausführend	1	100 %	

Aktivität NC-Programm übernehmen

Ressourcen			Name
Art	Anz.	Beteilig.	
			Fertigungsleitstand
Ausführend	1	100 %	

Aktivität NC-Programmierung

Ressourcen			Name
Art	Anz.	Beteilig.	
			NC Programmiersystem
Ausführend	1	100 %	

Aktivität Netzplanung

Ressourcen			Name
Art	Anz.	Beteilig.	
			PPS System
Ausführend	1	100 %	

```
Aktivität            PRP Lauf durchführen
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                       PPS System
  Ausführend       1         100 %

Aktivität            Planabweichung prüfen
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                       PPS System
  Ausführend       1         100 %

Aktivität            Projekt freigeben
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                       PPS System
  Ausführend       1         100 %

Aktivität            Prüfen ob NC Programm im FLS
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                       Fertigungsleitstand
  Ausführend       1         100 %

Aktivität            Prüfung Arbeitsplan
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                       PPS System
  Ausführend       1         100 %

Aktivität            Ressourcenbedarf spezifizieren
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                       Fertigungsleitstand
  Ausführend       1         100 %

Aktivität            Rückübertragung
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                       Fertigungsleitstand
  Ausführend       1         100 %

Aktivität            STL auf Vollständigkeit prüfen
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                       PPS System
  Ausführend       1         100 %

Aktivität            Simulationsplan übernehmen
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                       Fertigungsleitstand
  Ausführend       1         100 %

Aktivität            Simulationsplanung
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                       Fertigungsleitstand
  Ausführend       1         100 %

Aktivität            Standardbereich prüfen
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                       PPS System
  Ausführend       1         100 %

Aktivität            Strukturieren/Projekte bilden
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                       PPS System
  Ausführend       1         100 %

Aktivität            Stückliste prüfen
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                       PPS System
```

```
  Ausführend      1         100 %

Aktivität             Stücklisten erstellen
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                        PPS System
  Ausführend      1         100 %

Aktivität             Stücklisten verwalten
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                        PPS System
  Ausführend      1         100 %

Aktivität             Stücklisten/APL vervollständig
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                        PPS System
  Ausführend      1         100 %

Aktivität             Techn. u. terminl. Prüfung
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                        PPS System
  Ausführend      1         100 %

Aktivität             Teilestamm pflegen
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                        PPS System
  Ausführend      1         100 %

Aktivität             Termine abstimmen
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                        PPS System
  Ausführend      1         100 %

Aktivität             Updates einpflegen
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                        NC Programmiersystem
  Ausführend      1         100 %

Aktivität             Verkaufsangebot überführen
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                        Vertrieb KW/PY
  Ausführend      1         100 %

Aktivität             Wahrscheinlichk. gewichten
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                        Vertrieb KW/PY
  Ausführend      1         100 %

Aktivität             Wareneingang
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                        PPS System
  Ausführend      1         100 %

Aktivität             Werkzeugauslagerung
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                        Fertigungsleitstand
  Ausführend      1         100 %
                                        Werkzeugverwaltungssystem
  Ausführend      1         100 %

Aktivität             Werkzeugbereitstellung
                  Ressourcen            Name
  Art              Anz.  Beteilig.
                                        Werkzeugverwaltungssystem
  Ausführend      1         100 %
                                        Werkzeugvoreinstellgerät
  Ausführend      1         100 %
```

```
Aktivität            Werkzeugbruttobedarf ermitteln
                Ressourcen            Name
  Art            Anz.  Beteilig.
                                      Fertigungsleitstand
  Ausführend     1         100 %

Aktivität            Werkzeugdemontage
                Ressourcen            Name
  Art            Anz.  Beteilig.
                                      Werkzeugverwaltungssystem
  Ausführend     1         100 %

Aktivität            Werkzeugeinlagerung
                Ressourcen            Name
  Art            Anz.  Beteilig.
                           Fertigungsleitstand
 Ausführend     1         100 %
                           Werkzeugverwaltungssystem
             Ausführend     1         100 %

Aktivität            Werkzeugnettobedarf ermitteln
                Ressourcen            Name
  Art            Anz.  Beteilig.
                                      Fertigungsleitstand
  Ausführend     1         100 %
```

Datenelemente

```
Projekt:                        INTEGRATION (1)
Datenelement:                   %
Bericht:                        Datenelemente
Berichtsdatum:                  28-JAN-97
Ausgearbeitet von:              INCOMEDBA

Datenelement           Arbeitsplannummer
Parameter              Format:              Zeichenkette
                       Länge:               13
                       Nachkommastellen:    0

Objektspeicher              Netz
AVO-NC-Zuordnungen          Arbeitsplandaten lesen
Arbeitsplan vorhanden ?     Arbeitsplandaten lesen
Arbeitsplan vorhanden ?     Arbeitsplanübergabedatei lesen
Arbeitsplannummer           Arbeitsplanübergabedatei lesen
Auftrag vorhanden ?         Arbeitsplanübergabedatei lesen
Backup vorhanden ?          Arbeitsplanübergabedatei lesen
Datenabgleichsbedarf        Arbeitsplanübergabedatei lesen
Vorgänger vorhanden ?       Arbeitsplanübergabedatei lesen
Ähnlichen Arbeitsplan gefund Arbeitsplanübergabedatei lesen
Übernahmebefehl             Arbeitsplanübergabedatei lesen
AVO-NC-Zuordnungen          Auftragsdaten lesen
Arbeitsplan vorhanden ?     Auftragsdaten lesen
Arbeitsplan vorhanden ?     Auftragsübergabedatei lesen
Arbeitsplannummer           Auftragsübergabedatei lesen
Backup vorhanden ?          Auftragsübergabedatei lesen
Datenabgleichsbedarf        Auftragsübergabedatei lesen
Vorgänger gefunden?         Auftragsübergabedatei lesen
Ähnlicher APL gefunden?     Auftragsübergabedatei lesen

Datenelement           Arbeitsplanposition
Parameter              Format:              Ganze Zahl
                       Länge:               4
                       Nachkommastellen:    0

Objektspeicher                  Netz
AVO-NC-Zuordnungen              Arbeitsplandaten lesen
AVO-NC-Zuordnungen              Auftragsdaten lesen
Datenabgleichsbedarf            Auftragsdaten lesen

Datenelement           Auftragsnummer
Parameter              Format:              Zeichenkette
                       Länge:               13

                       Nachkommastellen:    0
```

Objektspeicher	Netz
Auftragsfreigabeaufforderung	Auftragsdaten lesen
Arbeitsplan vorhanden ?	Auftragsübergabedatei lesen
Arbeitsplannummer	Auftragsübergabedatei lesen
Auftrag vorhanden ?	Auftragsübergabedatei lesen
Auftragsfreigabeaufforderung	Auftragsübergabedatei lesen
Auftragsnummer	Auftragsübergabedatei lesen
Backup vorhanden ?	Auftragsübergabedatei lesen
Auftragsfreigabeaufforderung	Systeme koppeln
Rückmeldebedarf an PPS	Systeme koppeln

Datenelement	Boole'scher Wert	
Parameter	Format:	Ganze Zahl
	Länge:	1
	Nachkommastellen:	0

Objektspeicher	Netz
Arbeitsplan vorhanden ?	Arbeitsplandaten lesen
NC-Code verfügbar ?	Arbeitsplandaten lesen
Arbeitsplan vorhanden ?	Arbeitsplanübergabedatei lesen
Auftrag vorhanden ?	Arbeitsplanübergabedatei lesen
Backup vorhanden ?	Arbeitsplanübergabedatei lesen
Vorgänger vorhanden ?	Arbeitsplanübergabedatei lesen
Ähnlichen Arbeitsplan gefund	Arbeitsplanübergabedatei lesen
Arbeitsplan vorhanden ?	Auftragsdaten lesen
NC-Code vorhanden ?	Auftragsdaten lesen
Arbeitsplan vorhanden ?	Auftragsübergabedatei lesen
Auftrag vorhanden ?	Auftragsübergabedatei lesen
Backup vorhanden ?	Auftragsübergabedatei lesen
Vorgänger gefunden?	Auftragsübergabedatei lesen
Ähnlicher APL gefunden?	Auftragsübergabedatei lesen
Herkunftssystem	NC DIN Code lesen
Vollständigkeit	NC DIN Code lesen
Vorhanden ?	NC DIN Code lesen
NC-Programmverfügbarkeit	Systeme koppeln

Datenelement	Datei	
Parameter	Format:	Ganze Zahl
	Länge:	10
	Nachkommastellen:	0

Objektspeicher	Netz
Arbeitsplanübergabedatei	Arbeitsplandaten lesen
Abgewiesene Dateien	Arbeitsplanübergabedatei lesen
Arbeitsplanübergabedatei	Arbeitsplanübergabedatei lesen
Auftragsübergabedatei	Auftragsdaten lesen
Abgewiesene Dateien	Auftragsübergabedatei lesen
Auftragsübergabedatei	Auftragsübergabedatei lesen
Abgewiesene Dateien	NC DIN Code lesen
Differenz in NC-Programm	NC DIN Code lesen
NC Übergabedatei Input	NC DIN Code lesen
NC Übergabedatei Output	NC DIN Code schreiben
Arbeitsplanübergabedatei	Systeme koppeln
Auftragsübergabedatei	Systeme koppeln
Differenz in NC-Programm	Systeme koppeln
NC Übergabedatei Input	Systeme koppeln
NC Übergabedatei Output	Systeme koppeln
Rückmeldedatei (Input)	Systeme koppeln
Rückmeldedatei (Output)	Systeme koppeln
Übergabedatei FLS	Systeme koppeln

Datenelement	NC-Nummer	
Parameter	Format:	Zeichenkette
	Länge:	15
	Nachkommastellen:	0

Objektspeicher	Netz
AVO-NC-Zuordnungen	Arbeitsplandaten lesen
NC-Code verfügbar ?	Arbeitsplandaten lesen
AVO-NC-Zuordnungen	Auftragsdaten lesen
NC-Code vorhanden ?	Auftragsdaten lesen
Abgleichbedarf mit Vorgänger	NC DIN Code lesen
Herkunftssystem	NC DIN Code lesen
Neue Versionsnummer	NC DIN Code lesen
Programmnummer	NC DIN Code lesen
Vorhanden ?	NC DIN Code lesen

Vorhandene Version	NC DIN Code lesen
Aktuelle Versionsnummer	NC DIN Code schreiben
NC-Programmanforderung	NC DIN Code schreiben
Aktuelle Version	Systeme koppeln
NC-Programmanforderung	Systeme koppeln
NC-Programmverfügbarkeit	Systeme koppeln

Datenelement	Version	
Parameter	Format:	Zahl
	Länge:	7
	Nachkommastellen:	3

Objektspeicher	Netz
Abgleichbedarf mit Vorgänger	NC DIN Code lesen
Herkunftssystem	NC DIN Code lesen
Neue Versionsnummer	NC DIN Code lesen
Vorhandene Version	NC DIN Code lesen
Aktuelle Versionsnummer	NC DIN Code schreiben
Aktuelle Version	Systeme koppeln

Relationenschema des Schnittstellensystems

Tabellen Definitionen

Datenbank Trigger

Module

Datenbankschema in SQL DDL

```
REM
REM         This ORACLE7 command file was generated by Oracle
REM         Server Generator
REM         Version 5.5.8.0.0 on 03-JAN-97
REM
REM For application INTEGRATION version 1 database INTEGDB
REM
SET SCAN OFF
SPOOL ddlout.lst
REM TABLE CREATION
start       ddlout.tab
REM INDEX CREATION
start       ddlout.ind
REM CONSTRAINT CREATION
start       ddlout.con
REM TRIGGER CREATION
start       ddlout.trg
REM
REM         End of command file
REM
SPOOL OFF
```

Das Programm ddlout.tab

```
REM
REM         This ORACLE7 command file was generated by Oracle
REM         Server Generator
REM         Version 5.5.8.0.0 on 03-JAN-97
REM
REM For application INTEGRATION version 1 database INTEGDB
REM

CREATE TABLE zapls(
 arbeitspla VARCHAR2(13)      NOT NULL,
 herst_id      VARCHAR2(16)   NOT NULL,
 bak           INTEGER DEFAULT 0      NOT NULL
               CHECK ( bak BETWEEN 0 AND 1 ) ,
 ersteller     VARCHAR2(15)   NOT NULL,
 erstelldz     DATE           NOT NULL,
 gesperrt      INTEGER        DEFAULT 0      NOT NULL
               CHECK ( gesperrt BETWEEN 0 AND 1 ) ,
 kommentar  VARCHAR2(60)      NULL
);

CREATE TABLE zauftraege(
 ftgauftrag    VARCHAR2(13)   NOT NULL,
 status               INTEGER        DEFAULT 0      NOT NULL
                              CHECK ( status BETWEEN 0 AND 8 ) ,
 ersteller     VARCHAR2(15)   NOT NULL,
 erstelldz     DATE                  NOT NULL,
 stueck               INTEGER        NOT NULL,
 prioritaet    INTEGER        DEFAULT 0      NOT NULL,
 planfreig     INTEGER        DEFAULT 0      NOT NULL,
 fertfreig     INTEGER        DEFAULT 0      NOT NULL,
 bezeichng     VARCHAR2(60)   NOT NULL,
 projekt              VARCHAR2(9)    NOT NULL,
 herst_id             VARCHAR2(16)   NOT NULL,
 stueck_ist    INTEGER        DEFAULT 0      NOT NULL,
 prio_ist             INTEGER        DEFAULT 0      NOT NULL,
 plfrei_ist    INTEGER        DEFAULT 0      NOT NULL,
 fefrei_ist    INTEGER        DEFAULT 0      NOT NULL,
 mandant              INTEGER        NOT NULL,
 kommentar     VARCHAR2(60)   NULL,
 lieferung     DATE                  NULL,
 ftgstufe             VARCHAR2(3)    NULL,
 ablieferort          VARCHAR2(8)    NULL,
 apl_arbeitspla       VARCHAR2(13)   NOT NULL,
 apl_herst_id         VARCHAR2(16)   NOT NULL,
 apl_bak              INTEGER        NOT NULL
);
```

```
CREATE TABLE zavgtypen(
 avgtyp               VARCHAR2(3)    NOT NULL
);

CREATE TABLE zavgtypen_bearbeitungsplaetze(
 avgtyp_avgtyp        VARCHAR2(3)    NOT NULL,
 beapl_platznr        INTEGER        NOT NULL
);

CREATE TABLE zavo(
 apl_arbeitspla       VARCHAR2(13)   NOT NULL,
 apl_herst_id         VARCHAR2(16)   NOT NULL,
 apl_bak              INTEGER        NOT NULL,
 apl_index     INTEGER        NOT NULL,
 avg_nr               VARCHAR2(15)   NOT NULL,
 stueckzeit    NUMBER(10,0)   NOT NULL,
 stueck               INTEGER        NOT NULL,
 gesperrt             INTEGER        NOT NULL,
 messfreq             INTEGER        NOT NULL,
 kommentar     VARCHAR2(60)   NULL,
 automatik     INTEGER        NULL,
 ncnr_ncnr     VARCHAR2(15)   NULL,
 avgtyp_avgtyp        VARCHAR2(3)    NOT NULL
);

CREATE TABLE zavo_bearb(
 avo_apl_herst_id     VARCHAR2(16)   NOT NULL,
 avo_apl_index        INTEGER        NOT NULL,
 avo_apl_arbeitspla   VARCHAR2(13)   NOT NULL,
 avo_apl_bak          INTEGER        NOT NULL,
 bearb_typ_typ        VARCHAR2(3)    NOT NULL,
 bearb_nr             VARCHAR2(15)   NOT NULL
);

CREATE TABLE zavo_bea_1(
 avo_apl_herst_id     VARCHAR2(16)   NOT NULL,
 avo_apl_index        INTEGER        NOT NULL,
 avo_apl_arbeitspla   VARCHAR2(13)   NOT NULL,
 avo_apl_bak          INTEGER        NOT NULL,
 beapl_platznr        INTEGER        NOT NULL
);

CREATE TABLE zbeaarts(
 dtk                          VARCHAR2(3)    NOT NULL,
 editor               VARCHAR2(60)   NOT NULL
);

CREATE TABLE zbeadats(
 ncnrver_ver          NUMBER(7,3)    NOT NULL,
 ncnrver_ncnr_ncnr    VARCHAR2(15)   NOT NULL,
 beaart_dtk    VARCHAR2(3)    NOT NULL,
 datum                LONG RAW              NOT NULL
);

CREATE TABLE zbearb(
 typ_typ              VARCHAR2(3)    NOT NULL,
 nr                           VARCHAR2(15)   NOT NULL
);

CREATE TABLE zbearbeitungsplaetze(
 platznr                      INTEGER        NOT NULL,
 soflexname           VARCHAR2(24)   NOT NULL,
 tritonname           VARCHAR2(16)   NOT NULL,
 soflinsel_soflinsel  VARCHAR2(30)   NOT NULL
);

CREATE TABLE zbearbei_1(
 beapl_platznr        INTEGER        NOT NULL,
 station_station      VARCHAR2(20)   NOT NULL
);

CREATE TABLE zbeaver(
 ver                          NUMBER(7,3)    NOT NULL,
 bearb_typ_typ        VARCHAR2(3)    NULL,
 bearb_nr             VARCHAR2(15)   NULL,
 daten                LONG RAW              NOT NULL,
```

```
 ersteller     VARCHAR2(15)   NOT NULL,
 gesperrt              INTEGER        NOT NULL
                               CHECK ( gesperrt BETWEEN 0 AND 1 ) ,
 aktver                INTEGER        NOT NULL
                               CHECK ( aktver BETWEEN 0 AND 1 ) ,
 erstelldz     DATE                   NULL,
 kommentar     VARCHAR2(60)   NULL
);

CREATE TABLE zncnr(
 ncnr                  VARCHAR2(15)   NOT NULL
);

CREATE TABLE zncnrver(
 ver                           NUMBER(7,3)    NOT NULL,
 ncnr_ncnr     VARCHAR2(15)   NULL,
 ersteller     VARCHAR2(15)   NOT NULL,
 erstelldz     DATE                   NOT NULL,
 gesperrt              INTEGER        NOT NULL
                               CHECK ( gesperrt BETWEEN 0 AND 1 ) ,
 verwort               VARCHAR2(15)   NOT NULL,
 laufdr                NUMBER(10,0)   NOT NULL,
 stueck                INTEGER        NOT NULL,
 modfreig              INTEGER        NOT NULL
                               CHECK ( modfreig BETWEEN 0 AND 1 ) ,
 aktver                INTEGER        NOT NULL
                               CHECK ( aktver BETWEEN 0 AND 1 ) ,
 kommentar     VARCHAR2(60)   NULL
);

CREATE TABLE zplanungstermine(
 auftrag_ftgauftrag           VARCHAR2(13)   NOT NULL,
 avo_apl_herst_id             VARCHAR2(16)   NOT NULL,
 avo_apl_index                INTEGER        NOT NULL,
 avo_apl_arbeitspla           VARCHAR2(13)   NOT NULL,
 avo_apl_bak                  INTEGER        NOT NULL,
 anfang                       DATE                   NOT NULL,
 ende_frueh            DATE                   NOT NULL
);

CREATE TABLE zsoflinsel(
 soflinsel     VARCHAR2(30)   NOT NULL,
 inpfad                VARCHAR2(80)   NOT NULL,
 outpfad               VARCHAR2(80)   NOT NULL
);

CREATE TABLE zstatfreigs(
 ncnrver_ver           NUMBER(7,3)    NOT NULL,
 ncnrver_ncnr_ncnr     VARCHAR2(15)   NOT NULL,
 station_station       VARCHAR2(20)   NOT NULL,
 startfreig    INTEGER        NOT NULL
                               CHECK ( startfreig BETWEEN 0 AND 1 ) ,
 transfreig    INTEGER        NOT NULL
                               CHECK ( transfreig BETWEEN 0 AND 1 )
);

CREATE TABLE zstationen(
 station               VARCHAR2(20)   NOT NULL
);

CREATE TABLE ztyp(
 typ                           VARCHAR2(3)    NOT NULL,
 editor                VARCHAR2(60)   NOT NULL
);

CREATE TABLE zvibs(
 auftrag_ftgauftrag           VARCHAR2(13)   NOT NULL,
 status                       INTEGER        DEFAULT 0      NOT NULL
                                       CHECK ( status BETWEEN 0 AND 3 ) ,
 avo_apl_herst_id             VARCHAR2(16)   NULL,
 avo_apl_index                INTEGER        NULL,
 avo_apl_arbeitspla           VARCHAR2(13)   NULL,
 avo_apl_bak                  INTEGER        NULL,
 datum                        DATE                   NOT NULL
);
```

Das Programm ddlout.ind

```
REM
REM         This ORACLE7 command file was generated by Oracle
REM         Server Generator
REM         Version 5.5.8.0.0 on 03-JAN-97
REM
REM For application INTEGRATION version 1 database INTEGDB
REM

CREATE INDEX AUFTRAG__1 ON ZAUFTRAEGE
(           apl_arbeitspla ,
            apl_herst_id ,
            apl_bak ) PCTFREE 40;

CREATE INDEX AVGTYP_B_3 ON ZAVGTYPEN_BEARBEITUNGSPLAETZE
(           avgtyp_avgtyp ) PCTFREE  40;

CREATE INDEX AVGTYP_B_4 ON ZAVGTYPEN_BEARBEITUNGSPLAETZE
(           beapl_platznr ) PCTFREE  40 ;

CREATE INDEX AVO_APL__1 ON ZAVO
(           apl_arbeitspla ,
            apl_herst_id ,
            apl_bak ) PCTFREE 40;

CREATE INDEX AVO_AVGT_1 ON ZAVO
(           avgtyp_avgtyp ) PCTFREE  40;

CREATE INDEX AVO_BEAP_3 ON ZAVO_BEA_1
(           avo_apl_index ,
            avo_apl_bak ,
            avo_apl_herst_id ,
            avo_apl_arbeitspla ) PCTFREE    40;

CREATE INDEX AVO_BEAP_4 ON ZAVO_BEA_1
(           beapl_platznr ) PCTFREE  40;

CREATE INDEX AVO_BEAR_3 ON ZAVO_BEARB
(           avo_apl_index ,
            avo_apl_bak ,
            avo_apl_herst_id ,
            avo_apl_arbeitspla ) PCTFREE    40;

CREATE INDEX AVO_BEAR_4 ON ZAVO_BEARB
(           bearb_typ_typ ,
            bearb_nr ) PCTFREE       40;

CREATE INDEX AVO_NCNR_1 ON ZAVO
(           ncnr_ncnr ) PCTFREE      40;

CREATE INDEX BEADAT_B_1 ON ZBEADATS
(           beaart_dtk ) PCTFREE     40;

CREATE INDEX BEADAT_N_1 ON ZBEADATS
(           ncnrver_ver ,
            ncnrver_ncnr_ncnr ) PCTFREE     40;

CREATE INDEX BEAPL_SO_1 ON ZBEARBEITUNGSPLAETZE
(           soflinsel_soflinsel ) PCTFREE   40;

CREATE INDEX BEAPL_ST_3 ON ZBEARBEI_1
(           beapl_platznr ) PCTFREE  40;

CREATE INDEX BEAPL_ST_4 ON ZBEARBEI_1
(           station_station ) PCTFREE40;

CREATE INDEX BEARB_TY_1 ON ZBEARB
(           typ_typ ) PCTFREE 40;

CREATE INDEX BEAVER_B_1 ON ZBEAVER
(           bearb_typ_typ ,
            bearb_nr ) PCTFREE       40;

CREATE INDEX NCNRVER__1 ON ZNCNRVER
(           ncnr_ncnr ) PCTFREE      40;
```

```
CREATE INDEX PLT_AUFT_1 ON ZPLANUNGSTERMINE
(           auftrag_ftgauftrag ) PCTFREE    40;

CREATE INDEX PLT_AVO__1 ON ZPLANUNGSTERMINE
(           avo_apl_index ,
            avo_apl_bak ,
            avo_apl_herst_id ,
            avo_apl_arbeitspla ) PCTFREE    40;

CREATE INDEX STATFREI_3 ON ZSTATFREIGS
(           ncnrver_ver ,
            ncnrver_ncnr_ncnr ) PCTFREE     40;

CREATE INDEX STATFREI_4 ON ZSTATFREIGS
(           station_station ) PCTFREE40;

CREATE INDEX VIB_AUFT_1 ON ZVIBS
(           auftrag_ftgauftrag ) PCTFREE    40;

CREATE INDEX VIB_AVO__1 ON ZVIBS
(           avo_apl_index ,
            avo_apl_bak ,
            avo_apl_herst_id ,
            avo_apl_arbeitspla ) PCTFREE    40;
```

Das Programm ddlout.con

```
REM
REM         This ORACLE7 command file was generated by Oracle
REM         Server Generator
REM         Version 5.5.8.0.0 on 03-JAN-97
REM
REM For application INTEGRATION version 1 database INTEGDB
REM

ALTER TABLE ZAPLS ADD (
            CONSTRAINT APL_PK
            PRIMARY KEY (ARBEITSPLA,
            HERST_ID,
            BAK)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZAUFTRAEGE ADD (
            CONSTRAINT AUFTRAG_PK
            PRIMARY KEY (FTGAUFTRAG)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZAVGTYPEN ADD (
            CONSTRAINT AVGTYP_PK
            PRIMARY KEY (AVGTYP)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZAVGTYPEN_BEARBEITUNGSPLAETZE ADD (
            CONSTRAINT AVGTYP_B_2
            PRIMARY KEY (AVGTYP_AVGTYP,
            BEAPL_PLATZNR)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZAVO ADD (
            CONSTRAINT AVO_PK
            PRIMARY KEY (APL_INDEX,
            APL_BAK,
            APL_HERST_ID,
            APL_ARBEITSPLA)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZAVO_BEARB ADD (
            CONSTRAINT AVO_BEAR_2
            PRIMARY KEY (AVO_APL_INDEX,
```

```
            AVO_APL_HERST_ID,
            AVO_APL_ARBEITSPLA,
            AVO_APL_BAK,
            BEARB_TYP_TYP,
            BEARB_NR)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZAVO_BEA_1 ADD (
            CONSTRAINT AVO_BEAP_2
            PRIMARY KEY (AVO_APL_INDEX,
            AVO_APL_HERST_ID,
            AVO_APL_ARBEITSPLA,
            AVO_APL_BAK,
            BEAPL_PLATZNR)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZBEAARTS ADD (
            CONSTRAINT BEAART_PK
            PRIMARY KEY (DTK)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZBEADATS ADD (
            CONSTRAINT BEADAT_PK
            PRIMARY KEY (NCNRVER_VER,
            NCNRVER_NCNR_NCNR,
            BEAART_DTK)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZBEARB ADD (
            CONSTRAINT BEARB_PK
            PRIMARY KEY (TYP_TYP,
            NR)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZBEARBEITUNGSPLAETZE ADD (
            CONSTRAINT BEAPL_PK
            PRIMARY KEY (PLATZNR)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZBEARBEI_1 ADD (
            CONSTRAINT BEAPL_ST_2
            PRIMARY KEY (BEAPL_PLATZNR,
            STATION_STATION)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZBEAVER ADD (
            CONSTRAINT BEAVER_PK
            PRIMARY KEY (VER,
            BEARB_TYP_TYP,
            BEARB_NR)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZNCNR ADD (
            CONSTRAINT NCNR_PK
            PRIMARY KEY (NCNR)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZNCNRVER ADD (
            CONSTRAINT NCNRVER_PK
            PRIMARY KEY (VER,
            NCNR_NCNR)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZPLANUNGSTERMINE ADD (
            CONSTRAINT PLT_PK
            PRIMARY KEY (AVO_APL_INDEX,
            AUFTRAG_FTGAUFTRAG,
```

```
            AVO_APL_HERST_ID,
            AVO_APL_ARBEITSPLA,
            AVO_APL_BAK)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZSOFLINSEL ADD (
            CONSTRAINT SOFLINSEL_PK
            PRIMARY KEY (SOFLINSEL)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZSTATFREIGS ADD (
            CONSTRAINT STATFREIG_PK
            PRIMARY KEY (NCNRVER_VER,
            NCNRVER_NCNR_NCNR,
            STATION_STATION)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZSTATIONEN ADD (
            CONSTRAINT STATION_PK
            PRIMARY KEY (STATION)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZTYP ADD (
            CONSTRAINT TYP_PK
            PRIMARY KEY (TYP)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZVIBS ADD (
            CONSTRAINT VIB_PK
            PRIMARY KEY (STATUS,
            AUFTRAG_FTGAUFTRAG,
            AVO_APL_INDEX,
            AVO_APL_HERST_ID,
            AVO_APL_ARBEITSPLA,
            AVO_APL_BAK)
USING INDEX
PCTFREE     10 )/

ALTER TABLE ZAUFTRAEGE ADD (
            CONSTRAINT AUFTRAG_APL_FK
            FOREIGN KEY (     APL_ARBEITSPLA,
                              APL_HERST_ID,
                              APL_BAK)
            REFERENCES ZAPLS (        ARBEITSPLA,
                                      HERST_ID,
                                      BAK) )/

ALTER TABLE ZAVGTYPEN_BEARBEITUNGSPLAETZE ADD (
            CONSTRAINT AVGTYP_BEA_AVGTYP_FK
            FOREIGN KEY (     AVGTYP_AVGTYP)
            REFERENCES ZAVGTYPEN (    AVGTYP) )/

ALTER TABLE ZAVGTYPEN_BEARBEITUNGSPLAETZE ADD (
            CONSTRAINT AVGTYP_B_1
            FOREIGN KEY (     BEAPL_PLATZNR)
            REFERENCES ZBEARBEITUNGSPLAETZE (       PLATZNR) )/

ALTER TABLE ZAVO ADD (
            CONSTRAINT AVO_APL_FK
            FOREIGN KEY (     APL_ARBEITSPLA,
                              APL_HERST_ID,
                              APL_BAK)
            REFERENCES ZAPLS ( ARBEITSPLA,
                                      HERST_ID,
                                      BAK) )/

ALTER TABLE ZAVO ADD (
            CONSTRAINT AVO_NCNR_FK
            FOREIGN KEY (     NCNR_NCNR)
            REFERENCES ZNCNR ( NCNR) )/

ALTER TABLE ZAVO ADD (
```

```
            CONSTRAINT AVO_AVGTYP_FK
            FOREIGN KEY (     AVGTYP_AVGTYP)
            REFERENCES ZAVGTYPEN (    AVGTYP) )/

ALTER TABLE ZAVO_BEARB ADD (
            CONSTRAINT AVO_BEARB_AVO_FK
            FOREIGN KEY (     AVO_APL_INDEX,
                              AVO_APL_BAK,
                              AVO_APL_HERST_ID,
                              AVO_APL_ARBEITSPLA)
            REFERENCES ZAVO (  APL_INDEX,
                                          APL_BAK,
                                          APL_HERST_ID,
                                          APL_ARBEITSPLA) )/

ALTER TABLE ZAVO_BEARB ADD (
            CONSTRAINT AVO_BEAR_1
            FOREIGN KEY (     BEARB_TYP_TYP,
                              BEARB_NR)
            REFERENCES ZBEARB (       TYP_TYP,
                                          NR) )/

ALTER TABLE ZAVO_BEA_1 ADD (
            CONSTRAINT AVO_BEAPL_AVO_FK
            FOREIGN KEY (     AVO_APL_INDEX,
                              AVO_APL_BAK,
                              AVO_APL_HERST_ID,
                              AVO_APL_ARBEITSPLA)
            REFERENCES ZAVO (  APL_INDEX,
                                          APL_BAK,
                                          APL_HERST_ID,
                                          APL_ARBEITSPLA) )/

ALTER TABLE ZAVO_BEA_1 ADD (
            CONSTRAINT AVO_BEAP_1
            FOREIGN KEY (     BEAPL_PLATZNR)
            REFERENCES ZBEARBEITUNGSPLAETZE (PLATZNR) )/

ALTER TABLE ZBEADATS ADD (
            CONSTRAINT BEADAT_NCNRVER_FK
            FOREIGN KEY (     NCNRVER_VER,
                              NCNRVER_NCNR_NCNR)
            REFERENCES ZNCNRVER (     VER,
                                                NCNR_NCNR) )/

ALTER TABLE ZBEADATS ADD (
            CONSTRAINT BEADAT_BEAART_FK
            FOREIGN KEY (     BEAART_DTK)
            REFERENCES ZBEAARTS (     DTK) )/

ALTER TABLE ZBEARB ADD (
            CONSTRAINT BEARB_TYP_FK
            FOREIGN KEY (     TYP_TYP)
            REFERENCES ZTYP (  TYP) )/

ALTER TABLE ZBEARBEITUNGSPLAETZE ADD (
            CONSTRAINT BEAPL_SOFLINSEL_FK
            FOREIGN KEY (     SOFLINSEL_SOFLINSEL)
            REFERENCES ZSOFLINSEL (   SOFLINSEL) )/

ALTER TABLE ZBEARBEI_1 ADD (
            CONSTRAINT BEAPL_STAT_BEAPL_FK
            FOREIGN KEY (     BEAPL_PLATZNR)
            REFERENCES ZBEARBEITUNGSPLAETZE (PLATZNR) )/

ALTER TABLE ZBEARBEI_1 ADD (
            CONSTRAINT BEAPL_ST_1
            FOREIGN KEY (     STATION_STATION)
            REFERENCES ZSTATIONEN (   STATION) )/

ALTER TABLE ZBEAVER ADD (
            CONSTRAINT BEAVER_BEARB_FK
            FOREIGN KEY (     BEARB_TYP_TYP,
                              BEARB_NR)
            REFERENCES ZBEARB (       TYP_TYP,
                                          NR) )/
```

```
ALTER TABLE ZNCNRVER ADD (
            CONSTRAINT NCNRVER_NCNR_FK
            FOREIGN KEY (     NCNR_NCNR)
            REFERENCES ZNCNR ( NCNR) )/

ALTER TABLE ZPLANUNGSTERMINE ADD (
            CONSTRAINT PLT_AVO_FK
            FOREIGN KEY (     AVO_APL_INDEX,
                              AVO_APL_BAK,
                              AVO_APL_HERST_ID,
                              AVO_APL_ARBEITSPLA)
            REFERENCES ZAVO (  APL_INDEX,
                                          APL_BAK,
                                          APL_HERST_ID,
                                          APL_ARBEITSPLA) )/

ALTER TABLE ZPLANUNGSTERMINE ADD (
            CONSTRAINT PLT_AUFTRAG_FK
            FOREIGN KEY (     AUFTRAG_FTGAUFTRAG)
            REFERENCES ZAUFTRAEGE (   FTGAUFTRAG) )/

ALTER TABLE ZSTATFREIGS ADD (
            CONSTRAINT STATFREI_1
            FOREIGN KEY (     NCNRVER_VER,
                              NCNRVER_NCNR_NCNR)
            REFERENCES ZNCNRVER (     VER,
                                                  NCNR_NCNR) )/

ALTER TABLE ZSTATFREIGS ADD (
            CONSTRAINT STATFREI_2
            FOREIGN KEY (     STATION_STATION)
            REFERENCES ZSTATIONEN (   STATION) )/

ALTER TABLE ZVIBS ADD (
            CONSTRAINT VIB_AVO_FK
            FOREIGN KEY (     AVO_APL_INDEX,
                              AVO_APL_BAK,
                              AVO_APL_HERST_ID,
                              AVO_APL_ARBEITSPLA)
            REFERENCES ZAVO (  APL_INDEX,
                                          APL_BAK,
                                          APL_HERST_ID,
                                          APL_ARBEITSPLA) )/

ALTER TABLE ZVIBS ADD (
            CONSTRAINT VIB_AUFTRAG_FK
            FOREIGN KEY (     AUFTRAG_FTGAUFTRAG)
            REFERENCES ZAUFTRAEGE (   FTGAUFTRAG) )/
```

Das Programm ddlout.trg

```
REM
REM         This ORACLE7 command file was generated by Oracle
REM         Server Generator
REM         Version 5.5.8.0.0 on 03-JAN-97
REM
REM For application INTEGRATION version 1 database INTEGDB
REM

CREATE OR REPLACE TRIGGER avgplatztrig
AFTER DELETE
ON ZAVGTYPEN_BEARBEITUNGSPLAETZE
FOR EACH ROW
DECLARE
            c INTEGER;
BEGIN
            SELECT COUNT(*)
            INTO      c
            FROM      zavo_bea_1, zavo
            WHERE     avo_apl_arbeitspla = apl_arbeitspla AND
                      avo_apl_index = apl_index AND
                      avo_apl_bak = apl_bak AND
                      avo_apl_herst_id = apl_herst_id AND
                      zavo.avgtyp_avgtyp = :old.avgtyp_avgtyp AND
                      zavo_bea_1.beapl_platznr=:old.beapl_platznr;
```

```
            IF (c > 0) THEN
               raise_application_error(-20005,
                       'AVG-Platz Zuordnung kann wegen AVO-Platz
                       Zuordnungen nicht geloest werden' );
            END IF;
END;
/

CREATE OR REPLACE TRIGGER beatrigger
AFTER INSERT OR DELETE OR UPDATE
ON ZBEAVER
DECLARE
            a INTEGER;
            b INTEGER;
BEGIN
            SELECT    MAX(sum(aktver)), MIN(sum(aktver))
            INTO      a, b
            FROM      zbeaver group BY bearb_nr;
            IF(a <> 1) OR(b<>1) THEN
               raise_application_error(-20001,
                       'Integritaetsverletzung IN Tabelle
                       ZBEAVER');
            END IF;
END;
/

CREATE OR REPLACE TRIGGER nctrigger
AFTER INSERT OR DELETE OR UPDATE
ON ZNCNRVER
DECLARE
            a INTEGER;
            b INTEGER;
BEGIN
            SELECT    MAX(sum(aktver)), MIN(sum(aktver))
            INTO      a, b
            FROM      zncnrver group BY ncnr_ncnr;
            IF(a <> 1) OR(b<>1) THEN
               raise_application_error(-20001,
                       'Integritaetsverletzung in Tabelle
                       ZNCNRVER');
            END IF;
END;
/

CREATE OR REPLACE TRIGGER platzavo
AFTER INSERT OR UPDATE
ON ZAVO_BEA_1
DECLARE
            x INTEGER;
            y INTEGER;
            z INTEGER;
BEGIN
            SELECT    COUNT(*)
            INTO              y
            FROM      zavo_bea_1;
            IF(y > 0) THEN
               SELECT  MAX(COUNT(distinct soflinsel))
               INTO           z
               FROM           zavo_bea_1,
                              zbearbeitungsplaetze,
                              zsoflinsel
               WHERE   zbearbeitungsplaetze.platznr =
                                     zavo_bea_1.beapl_platznr AND
                              zsoflinsel.soflinsel =
                                     zbearbeitungsplaetze.
                                     soflinsel_soflinsel
               GROUP BY       avo_apl_arbeitspla,
                              avo_apl_index,
                              avo_apl_bak,
                              avo_apl_herst_id;
               IF(z > 1) THEN
                      raise_application_error(-20002,
                              'Mehrere Fertigungsinseln
                               angesprochen');
               END IF;
            END IF;
            SELECT    MAX(COUNT(b.beapl_platznr))
```

```
            INTO      x
            FROM      zavo a,
                      zavo_bea_1 b
            WHERE     a.apl_arbeitspla = b.avo_apl_arbeitspla AND
                      a.apl_herst_id = b.avo_apl_herst_id AND
                      a.apl_bak = b.avo_apl_bak AND
                      a.apl_index = b.avo_apl_index AND
                      b.beapl_platznr NOT IN(
               SELECT avgtypen_bearbeitungsplaetze.beapl_platznr
               FROM          zavo c,
                             zavgtypen,
                             zavgtypen_bearbeitungsplaetze
               WHERE  c.avgtyp_avgtyp = zavgtypen.avgtyp AND
                             zavgtypen.avgtyp =
                                    zavgtypen_bearbeitungsplaetze.
                                    avgtyp_avgtyp AND
                             a.apl_arbeitspla = c.apl_arbeitspla AND
                             a.apl_index = c.apl_index AND
                             a.apl_bak = c.apl_bak AND
                             a.apl_herst_id = c.apl_herst_id)
            GROUP BY  a.apl_arbeitspla,
                      a.apl_herst_id,
                      a.apl_bak,
                      a.apl_index;
            IF(x > 0) THEN
               raise_application_error(-20004,
                      'Integritaetsverletzung Tabelle
                      zavo_bea_1');
            END IF;
END;
/

CREATE OR REPLACE TRIGGER beaplatz_trigger
AFTER INSERT OR DELETE OR UPDATE
ON ZBEARBEITUNGSPLAETZE
DECLARE
            a INTEGER;
            b INTEGER;
            c INTEGER;
BEGIN
            SELECT    COUNT(distinct soflexname)
            INTO            a
            FROM      zbearbeitungsplaetze;
            SELECT    COUNT(distinct tritonname)
            INTO      b
            FROM      zbearbeitungsplaetze;
            SELECT    COUNT(*)
            INTO      c
            FROM      zbearbeitungsplaetze;
            IF(a <> c) THEN
               raise_application_error(-20002, 'Fehler ! Soflex-
               Platz schon vorhanden');
            END IF;
            IF(b <> c) THEN
               raise_application_error(-20002, 'Fehler ! Triton-
               Platz schon vorhanden');
            END IF;
END;
```

Basic-Programm zum Einlesen der Arbeitspläne und Fertigungsaufträge

```
'=========================================================
' (C) ABB Kraftwerke AG
'
' Autor : Joerg Evermann
'         erstellt im Rahmen der Diplomhausarbeit,
'         Oktober 1996
'
'=========================================================
'=========================================================
' Pruefen ob Dateien vorhanden sind
'=========================================================
Sub bearb ()

    Dim pref As String
```

```
    Dim I As Integer
    Dim aktaup As String

    I = 0
    While (StrComp(Right$(FileListBox.List(I), 4), ".AUP", 1) <> 0) And
           (FileListBox.ListCount >= I)
           I = I + 1
    Wend

    If (FileListBox.ListCount >= I) Then
    'If (StrComp(Right$(NCANZEIGE.FLB1.List(i), 3), ".NC", 1) = 0) Then
           aktaup = FileListBox.List(I)
    Else
           Exit Sub
    End If

    pref = Left$(aktaup, InStr(1, aktaup, ".") - 1)

    AktuellDatei.Caption = globalspfad & aktaup
    'jetzt alle files in die datenbank schreiben
    ReadIntoDb (pref)
    'die Datei dann loeschen
    Kill globalspfad & aktaup
    'und die anzeige wegmachen
    AktuellDatei.Caption = ""

    Exit Sub

End Sub

'============================================================
' Dieses Unterprogramm wird beim Programmstart aufgerufen
' Anmelden an die Datenbank und Lesen der Parameterdatei
'============================================================
Sub Form_Load ()

  On Error GoTo Form_Load_Err
  ' Log onto database
  Call ldLogon
  ' Display the connection string
  'pnlUser.Caption = "  User:  " & gstrUserName & "@" & gstrDatabaseName

  ' Display the date and time
  'Call lgStatusDate(Me!pnlDate)
  ' Display the status bar
  'mnuStatusBar.Checked = True
  ' Centralize this form
  Call lgCentralizeForm(Me, Me)

  ' To avoid of MDI Child default window sizing, set the form size
   If Me.MDIChild Then
      Me.Height = 2235 + gHEIGHT_ADJUSTMENT
      Me.Width = 5395 + gWIDTH_ADJUSTMENT
   End If

  ' Initialise form state with respect to zone coordination
  'Call TYP_Clear(False, True)
  mbAutoQuery = True

  ' Ensure that the list is in the foreground
  'fraGotoRec.ZOrder

  ' Create dynaset for DML operations
  'If Not TYP_CreDysBase() Then
  '  Call lgMsg(glmtFORM_NOT_USABLE, gMG_SEV_ERROR)
  '  Exit Sub
  'End If

  Dim Msg, Success    ' Declare variables
  Dim KName$, Ret$, SName$

  SName$ = "AUPLOOP" ' WIN.INI section name.
  KName$ = "Suchpfad" '
  Ret$ = String$(255, 0)  ' Initialize return string.
  ' Call Windows Kernel DLL.
  Success = GetPrivateProfileString(SName$, KName$, "c:\", Ret$, Len(Ret$), "integ.ini")
  Ret = Left$(Ret$, InStr(1, Ret$, Chr(0)) - 1)
```

```
  globalspfad$ = Ret$
  verz = GetPrivateProfileInt(SName$, "Verzoegerung", 1000, "integ.ini")
  Ret$ = String$(255, 0)  ' Initialize return string.
  Success = GetPrivateProfileString(SName$, "Fehlerpfad", "c:\", Ret$, Len(Ret$),
            integ.ini")
  Ret = Left$(Ret$, InStr(1, Ret$, Chr(0)) - 1)
  globalfehlerpfad = Ret$
  Ret$ = String$(255, 0)  ' Initialize return string.
  Success = GetPrivateProfileString(SName$, "Logdatei", "c:\ncloop.log", Ret$,
            Len(Ret$), "integ.ini")
  Ret = Left$(Ret$, InStr(1, Ret$, Chr(0)) - 1)
  globallogfile = Ret$
  Me.Caption = "AUP-Loop auf " & globalspfad & " alle " & Str$(verz) & " msec"

  FileListBox.Path = globalspfad
  SuchPfadText.Caption = "Verzeichnis: " & FileListBox.Path
  Timer1.Interval = verz  ' Set timer interval.

  Exit Sub

Form_Load_Err:
   Call lgMsgVBError("MENUE.FRM", "Form_Load", Err, Error)
  Exit Sub

End Sub

'============================================================
' Beenden des Programms
'============================================================
Sub LOOPEND_Click ()
    End
End Sub

'============================================================
' Auftrag und Arbeitsplan einlesen
'============================================================
Sub ReadIntoDb (prefix As String)

    On Error GoTo fehlerbehandlung

    Dim I As Integer
    Dim theSet As Object
    Dim theStatSet As Object
    Dim theAktSet As Object
    Dim ftgnr As String
    Dim herstid As String
    Dim gesperrt As Integer
    Dim FileNum As Integer
    Dim aplnr As String

    On Error GoTo fehlerbehandlung
    gsession.DbBeginTrans

    ' Datei oeffnen
    FileNum = FreeFile
    filename = globalspfad & prefix & ".aup"
    Open filename For Input As #FileNum
    Line Input #FileNum, theLine
    On Error GoTo fehlerbehandlung

    ' Erste Zeile Auftragsnummer lesen'
    While (StrComp(Left$(theLine, 15), "FTGAUFTRAG.....") <> 0) And (Not EOF(FileNum))
            Line Input #FileNum, theLine
    Wend
    If (EOF(FileNum)) Then
            ErrFnum = FreeFile
            Open globallogfile For Append Access Write As #ErrFnum
            Print #ErrFnum, "***ERROR*** " & Date$ & " " & Time$
            Print #ErrFnum, "   DateiName     : " & prefix
            Print #ErrFnum, "   Bemerkung     : Kann Auftragsnummer nicht finden!"
            Close
            gsession.DbRollBack
            Exit Sub
    End If
    ftgnr = Left$(Trim(Right$(theLine, Len(theLine) - 16)), 13)
```

```
    Set theSet = gdatabase.DbCreateDynaset("select ftgauftrag from zauftraege where
            ftgauftrag = '" & ftgnr & "'", 0&)
    If (Not theSet.EOF()) Then
            'Fertigungsauftrag auf Arbeitsplan freigegeben !
            ErrFnum = FreeFile
            Open globallogfile For Append Access Write As #ErrFnum
            Print #ErrFnum, "***ERROR*** " & Date$ & " " & Time$
            Print #ErrFnum, "   DateiName     : " & prefix
            Print #ErrFnum, "   Bemerkung     : Kann Fertigungsauftrag " &
               ftgnr & " nicht lesen, Fertigungsauftrag existiert !"
            Close
            gsession.DbRollBack
            Exit Sub
    End If

    While (StrComp(Left$(theLine, 15), "ARBEITSPLAN....") <> 0) And (Not EOF(FileNum))
            Line Input #FileNum, theLine
    Wend
    If (EOF(FileNum)) Then
            ErrFnum = FreeFile
            Open globallogfile For Append Access Write As #ErrFnum
            Print #ErrFnum, "***ERROR*** " & Date$ & " " & Time$
            Print #ErrFnum, "   DateiName     : " & prefix
            Print #ErrFnum, "   Bemerkung     : Kann Arbeitsplannummer finden!"
            Close
            gsession.DbRollBack
            Exit Sub
    End If

    aplnr = Left$(Trim(Right$(theLine, Len(theLine) - 16)), 13)

    Set theSet = gdatabase.DbCreateDynaset("select ftgauftrag from zauftraege where
            apl_arbeitspla = '" & aplnr & "'", 0&)
    If (Not theSet.EOF()) Then
            'Fertigungsauftrag auf Arbeitsplan freigegeben !
            ErrFnum = FreeFile
            Open globallogfile For Append Access Write As #ErrFnum
            Print #ErrFnum, "***ERROR*** " & Date$ & " " & Time$
            Print #ErrFnum, "   DateiName     : " & prefix
            Print #ErrFnum, "   Bemerkung     : Kann Arbeitsplan " & aplnr
               & " nicht einlesen, Fertigungsauftrag existiert !"
            Close
            gsession.DbRollBack
            Exit Sub
    End If

    Set theSet = gdatabase.DbCreateDynaset("select ftgauftrag from zauftraege where
            apl_bak = 0 and apl_herst_id = '" & herst_id & "' and apl_arbeitspla = '" &
            arbeitsplan & "' and ftgauftrag <> '" & ftgnr & "'", 0&)
    If (theSet.EOF()) Then
            'es gibt keinen auftrag zu diesem arbeitsplan, ausser dem aktuellen
            'jetzt den arbeitsplan lesen
            If Not ReadAPL(aplnr, prefix & ".aup", FileNum, ftgnr, False) Then
                'der Arbeitsplan ist nicht richtig gelesen worden
                GoTo fehlerbehandlung
            Else
                If Verfuegbarkeitspruefung(aplnr) Then
                    Call Freigeben(ftgnr, aplnr)
                End If
            End If
    End If

    gsession.DbCommitTrans

    ErrFnum = FreeFile
    Open globallogfile For Append Access Write As #ErrFnum
    Print #ErrFnum, "***SUCCESS*** " & Date$ & " " & Time$
    Print #ErrFnum, "   Dateien #     : " & prefix
    Print #ErrFnum, ""
    Close #ErrFnum
    Close
    Exit Sub

fehlerbehandlung:

    Close
```

```
    SrcName = globalspfad & prefix & ".aup"
    DestName = globalfehlerpfad & prefix & ".aup"
    FileCopy SrcName, DestName

    ErrFnum = FreeFile
    Open globallogfile For Append Access Write As #ErrFnum
    Print #ErrFnum, "***ERROR*** " & Date$ & " " & Time$
    Print #ErrFnum, "   VB Fehler #   : " & Str$(Err)
    Print #ErrFnum, "   VB Fehler     : " & Error$(Err)
    Print #ErrFnum, "   Oracle Fehler : " & gdatabase.LastServerErrText
    Print #ErrFnum, "   Datei #       : " & prefix
    Print #ErrFnum, "   Arbeitsplan # : " & aplnr
    Print #ErrFnum, "   Fehlerhafte Zl: " & theLine
    Print #ErrFnum, ""
    Close #ErrFnum
    gdatabase.LastServerErrReset

    gsession.DbRollBack
    Exit Sub

End Sub

'=========================================================
' Wird durch den Timer regelmaessig aufgerufen, loest die
' Ueberpruefung des Verzeichnisses auf Dateien aus.
'=========================================================
Sub Timer1_Timer ()
    FileListBox.Refresh
    If FileListBox.ListCount > 0 Then
            bearb
    End If
End Sub

'=========================================================
' Liest eine Arbeitsplanposition mit Terminen aus Datei
'=========================================================
Function avolesen(filename As String, aplnr As String, herstid As String, FileNum As
            Integer, aupnr As String, Auftraglesen As Integer) As Integer

    Dim srcname As String
    Dim destname As String
    Dim ErrFnum As Integer
    Dim theLine As String
    Dim theSet As Object
    Dim aplindex As Long
    Dim ncnr As String

    avolesen = False
    On Error GoTo fehlerbehandlung2

    theLine = LTrim(RTrim(theLine))

    Set theSet = gDatabase.DbCreateDynaset("select messfreq, apl_arbeitspla,
            apl_herst_id, apl_bak, apl_index, avg_nr, stueckzeit, stueck, gesperrt,
            automatik, kommentar, ncnr_ncnr, avgtyp_avgtyp from zavo", 0&)
    theSet.DbAddNew
    theSet.Fields("apl_arbeitspla").Value = aplnr
    theSet.Fields("apl_bak").Value = 0
    theSet.Fields("apl_herst_id").Value = herstid

    While (Left$(theLine, 6) <> "AVOEND") And (Not EOF(FileNum)) And (Left$(theLine, 11)
            <> "TERMINBEGIN")

        Line Input #FileNum, theLine

        Select Case Left$(theLine, 15)
            Case "APLINDEX......."
                theSet.Fields("apl_index").Value = Left$(Trim(right$(theLine,
                            Len(theLine) - 16)), 4)
                aplindex = Val(Left$(Trim(right$(theLine, Len(theLine) - 16)), 4))
            Case "AVG_NR........."
                theSet.Fields("avg_nr").Value = Left$(Trim(right$(theLine,
                            Len(theLine) - 16)), 15)
            Case "NCNR.........."
                ncnr = Left$(Trim(right$(theLine, Len(theLine) - 16)), 15)
            Case "KOMMENTAR......"
                theSet.Fields("kommentar").Value = Left$(Trim(right$(theLine,
```

```
                        Len(theLine) - 16)), 60)
            Case "STUECK........."
                theSet.Fields("stueck").Value = Left$(Trim(right$(theLine,
                        Len(theLine) - 16)), 10)
            Case "GESPERRT......."
                theSet.Fields("gesperrt").Value = Left$(Trim(right$(theLine,
                        Len(theLine) - 16)), 1)
            Case "AUTOMATIK......"
                theSet.Fields("automatik").Value = Left$(Trim(right$(theLine,
                        Len(theLine) - 16)), 1)
            Case "MESSFREQ......."
                theSet.Fields("messfreq").Value = Left$(Trim(right$(theLine,
                        Len(theLine) - 16)), 10)
            Case "AVGTYP........."
                theSet.Fields("avgtyp_avgtyp").Value = Left$(Trim(right$(theLine,
                        Len(theLine) - 16)), 3)
            Case "STUECKZEIT....."
                theLine = Trim$(right$(theLine, Len(theLine) - 16))
                theSet.Fields("stueckzeit").Value = Val(Mid$(theLine, 1, 4)) * 60 * 60
                        * 24 + Val(Mid$(theLine, 6, 2)) * 60 * 60 +
                        Val(Mid$(theLine, 9, 2)) * 60 + Val(Mid$(theLine, 12, 2))
        End Select

    Wend
    theSet.DbUpdate

    If (ncnr <> "") Then
        If (ncnrexists(ncnr) = True) Then
            gDatabase.DbExecuteSQL ("update zavo set ncnr_ncnr = '" & ncnr & "' where
                    apl_arbeitspla = '" & aplnr & "' and apl_bak = 0")
        Else
            gDatabase.DbExecuteSQL ("insert into zncnr values ('" & ncnr & "')")
                gDatabase.DbExecuteSQL ("insert into zncnrver (ver, ncnr_ncnr,
                    ersteller, erstelldz, gesperrt, verwort, laufdr, stueck,
                    modfreig, aktver, kommentar) values (0, '" & ncnr & "', 'PRODIN',
                    SYSDATE, 1, 'PRODIN', 0, 0, 0, 1, 'PRODIN SYSTEM')")
            gDatabase.DbExecuteSQL ("insert into zbeadats (ncnrver_ver,
                    ncnrver_ncnr_ncnr, beaart_dtk, datum) values (0, '" & ncnr
                    & "', 'NC', hextoraw(32))")
            gDatabase.DbExecuteSQL ("update zavo set ncnr_ncnr = '" & ncnr & "' where
                    apl_arbeitspla = '" & aplnr & "' and apl_bak = 0")
        End If
    End If

    Do
        If (Left$(theLine, 11) = "TERMINBEGIN") And (Auftraglesen) Then
            'lesen wir ein fhmzuordnung
            Select Case terminlesen(filename, aplnr, herstid, aplindex, aupnr, FileNum)
                Case -1:
                    'alles passt
                Case 1:
                    'weder nummer noch typ sind da
                    ErrFnum = FreeFile
                    Open globallogfile For Append Access Write As #ErrFnum
                    Print #ErrFnum, "***ERROR*** " & Date$ & " " & Time$
                    Print #ErrFnum, "   Datei #       : " & filename
                    Print #ErrFnum, "   Arbeitsplan # : " & aplnr
                    Print #ErrFnum, "    APLIndex #   : " & aplindex
                    Print #ErrFnum, "   Kommentar     : Es wurde ein ungueltiger Termin
                                    gelesen"
                    Print #ErrFnum, ""
                    Close #ErrFnum
                    avolesen = False
                    Exit Function
            End Select
            'wenn wir nicht am dateiende sind
            If (Not EOF(FileNum)) Then
                'lesen wir noch eine zeile
                Line Input #FileNum, theLine
            End If
        End If

        If (Left$(theLine, 10) = "PLATZBEGIN") Then
            'lesen wir ein fhmzuordnung
            Select Case platzlesen(filename, aplnr, herstid, aplindex, FileNum)
                Case -1:
                    'alles passt
```

```
                Case 1:
                    'weder nummer noch typ sind da
                    ErrFnum = FreeFile
                    Open globallogfile For Append Access Write As #ErrFnum
                    Print #ErrFnum, "***ERROR*** " & Date$ & " " & Time$
                    Print #ErrFnum, "   Datei #       : " & filename
                    Print #ErrFnum, "   Arbeitsplan # : " & aplnr
                    Print #ErrFnum, "    APLIndex #   : " & aplindex
                    Print #ErrFnum, "   Kommentar     : Es wurde ein Platz gelesen, der
                                     nicht existiert!"
                    Print #ErrFnum, ""
                    Close #ErrFnum
                    avolesen = False
                    Exit Function
            End Select
            'wenn wir nicht am dateiende sind
            If (Not EOF(FileNum)) Then
                Line Input #FileNum, theLine
            End If
        End If

        If (Left$(theLine, 8) = "FHMBEGIN") Then
            'lesen wir ein fhmzuordnung
            Select Case fhmlesen(filename, aplnr, herstid, aplindex, FileNum)
                Case -1:
                    'alles passt
                Case 0:
                    'fhmnummer ist nicht da
                    gDatabase.DbExecuteSQL ("update zapls set gesperrt = 1 where
                                    arbeitspla = '" & aplnr & "' and bak = 0")
                Case 1:
                    'weder nummer noch typ sind da
                    ErrFnum = FreeFile
                    Open globallogfile For Append Access Write As #ErrFnum
                    Print #ErrFnum, "***ERROR*** " & Date$ & " " & Time$
                    Print #ErrFnum, "   Datei #       : " & filename
                    Print #ErrFnum, "   Arbeitsplan # : " & aplnr
                    Print #ErrFnum, "    APLIndex #   : " & aplindex
                    Print #ErrFnum, "   Kommentar     : Es wurde ein FHM gelesen, dessen
                                     Typ nicht existiert!"
                    Print #ErrFnum, ""
                    Close #ErrFnum
                    avolesen = False
                    Exit Function
            End Select
            'wenn wir nicht am dateiende sind
            If (Not EOF(FileNum)) Then
                Line Input #FileNum, theLine
            End If
        End If
    Loop While (Not EOF(FileNum)) And (Left$(theLine, 6)<> "AVOEND")

    avolesen = True
    Exit Function

fehlerbehandlung2:
    Close

    srcname = globalspfad & filename
    destname = globalfehlerpfad & filename
    FileCopy srcname, destname

    ErrFnum = FreeFile
    Open globallogfile For Append Access Write As #ErrFnum
    Print #ErrFnum, "***ERROR*** " & Date$ & " " & Time$
    Print #ErrFnum, "   VB Fehler #   : " & Str$(Err)
    Print #ErrFnum, "   VB Fehler     : " & Error$(Err)
    Print #ErrFnum, "   Oracle Fehler : " & gDatabase.LastServerErrText
    Print #ErrFnum, "   Datei #       : " & filename
    Print #ErrFnum, "   Arbeitsplan # : " & aplnr
    Print #ErrFnum, "    APLIndex #   : " & aplindex
    Print #ErrFnum, "   Fehlerhafte Zl: " & theLine
    Print #ErrFnum, ""
    Close #ErrFnum
    gDatabase.LastServerErrReset
    avolesen = False
    Exit Function
```

```
End Function

'=========================================================
' Gleicht die Fertigungshilfsmittelzuordnungen mit der
' Vorgaengerversion ab
'=========================================================
Function FHMAbgleich(aplnr As String, herstid As String) As Integer
    Dim stmtxt As String
    Dim ErrFnum As Integer
    On Error GoTo fehler_fhmabgleich
    FHMAbgleich = False

    stmtxt = "insert into zavo_bearb ( "
    stmtxt = stmtxt & "avo_apl_index, avo_apl_arbeitspla,avo_apl_herst_id, avo_apl_bak,
            bearb_typ_typ, bearb_nr )"
    stmtxt = stmtxt & " select avo_apl_index, avo_apl_arbeitspla, avo_apl_herst_id, 0,
            bearb_typ_typ, bearb_nr"
    stmtxt = stmtxt & " from zavo_bearb avo"
    stmtxt = stmtxt & " where avo.avo_apl_arbeitspla = '" & aplnr & "' and"
    stmtxt = stmtxt & " avo.avo_apl_bak = 1 and"
    stmtxt = stmtxt & " avo.avo_apl_herst_id = '" & herstid & "' and"
    stmtxt = stmtxt & " avo.avo_apl_index in ( ("
    stmtxt = stmtxt & " select apl_index from zavo where"
    stmtxt = stmtxt & " zavo.apl_arbeitspla = avo.avo_apl_arbeitspla and"
    stmtxt = stmtxt & " zavo.apl_herst_id = avo.avo_apl_herst_id and"
    stmtxt = stmtxt & " zavo.apl_bak = 0 ) minus ("
    stmtxt = stmtxt & " select avo_apl_index from zavo_bearb where"
    stmtxt = stmtxt & " zavo_bearb.avo_apl_arbeitspla = avo.avo_apl_arbeitspla and"
    stmtxt = stmtxt & " zavo_bearb.avo_apl_herst_id = avo.avo_apl_herst_id and"
    stmtxt = stmtxt & " zavo_bearb.avo_apl_bak = 0 ) )"

    FHMAbgleich = True

    Exit Function

fehler_fhmabgleich:

    ErrFnum = FreeFile
    Debug.Print stmtxt
    Open globallogfile For Append Access Write As #ErrFnum
    Print #ErrFnum, "***ERROR*** " & Date$ & " " & Time$
    Print #ErrFnum, "   Arbeitsplan # : " & aplnr
    Print #ErrFnum, "   Kommentar     : Fehler beim FHM Abgleich mit altem Arbeitsplan,"
    Print #ErrFnum, "                 : siehe folgende Fehlermeldung"
    Print #ErrFnum, ""
    Close #ErrFnum

    Exit Function
End Function

'=========================================================
' Liest die FHM Zuordnungen zu einer Arbeitsplanposition
'=========================================================
Function fhmlesen(filename As String, aplnr As String, herstid As String, aplindex As
            Long, FileNum As Integer) As Integer

    Dim theLine As String
    Dim fhmtyp As String
    Dim fhmnummer As String
    Dim zuord As Integer
    Dim ErrFnum As Integer
    Dim theTestSet As Object
    Dim ncnr As String

    fhmlesen = 0
    On Error GoTo fehlerbehandlung3

    theLine = LTrim(RTrim(theLine))
    zuord = 0

    While (Left$(theLine, 6) <> "FHMEND") And (Not EOF(FileNum))

        Line Input #FileNum, theLine
        Select Case Left$(theLine, 15)
            Case "FHMNUMMER......"
                fhmnummer = Left$(Trim(right$(theLine, Len(theLine) - 16)), 15)
            Case "FHMTYP........."
```

```
                fhmtyp = Left$(Trim(right$(theLine, Len(theLine) - 16)), 3)
        End Select
    Wend

    Set theTestSet = gDatabase.DbCreateDynaset("select nr from zbearb where nr = '" &
                                    fhmnummer & "' and typ_typ = '" & fhmtyp & "'", 0&)
    If (Not theTestSet.EOF()) Then
        'nummer und typ sind da
        gDatabase.DbExecuteSQL ("insert into zavo_bearb (avo_apl_index,
                    avo_apl_arbeitspla, avo_apl_herst_id, avo_apl_bak, bearb_typ_typ,
                    bearb_nr) values (" & Left$(Trim(aplindex), 4) & ", '" & aplnr &
                    "', '" & herstid & "', 0, '" & fhmtyp & "', '" & fhmnummer & "')")
        fhmlesen = -1
    Else
        Set theTestSet = gDatabase.DbCreateDynaset("select typ from ztyp where typ = '"
            & fhmtyp & "'", 0&)
        If (Not theTestSet.EOF()) Then
            'typ ist da, nummer nicht
            gDatabase.DbExecuteSQL ("insert into zbearb (typ_typ, nr) values ('" &
                    fhmtyp & "', '" & fhmnummer & "')")
            gDatabase.DbExecuteSQL ("insert into zbeaver (ver, bearb_typ_typ, bearb_nr,
                    daten, ersteller, gesperrt, aktver, erstelldz, kommentar) values
                    (0, '" & fhmtyp & "', '" & fhmnummer & "', hextoraw (32),
                    'PRODINSYS', 1, 1, SYSDATE, 'PRODIN SYSTEM')")
            'gdatabase.DbExecuteSQL ("insert into zbeaver (bearb_nr, ver, daten,
                    ersteller, gesperrt, erstelldz, kommentar, aktver)
                    values ('" & fhmtyp & "', '" & fhmnummer & "', 0.0,
                    hextoraw('00'), 'DUMMY', 1, '01-JAN-80', 'DUMMY', 1)")
            gDatabase.DbExecuteSQL ("insert into zavo_bearb (avo_apl_index,
                    avo_apl_arbeitspla, avo_apl_herst_id, avo_apl_bak, bearb_typ_typ,
                    bearb_nr) values (" & Left$(Trim(aplindex), 4) & ", '" & aplnr &
                    "', '" & herstid & "', 0, '" & fhmtyp & "', '" & fhmnummer & "')")
            fhmlesen = 0
        Else
            'weder nummer noch typ gefunden, jetzt
            'richtiges problem!
            'hier muss ein vb fehler ausgeloest werden
            'und der ganze arbeitsplan muss weg
            fhmlesen = 1
        End If
    End If
    Exit Function

fehlerbehandlung3:

    ErrFnum = FreeFile
    Open globallogfile For Append Access Write As #ErrFnum
    Print #ErrFnum, "***WARNING*** " & Date$ & " " & Time$
    Print #ErrFnum, "   VB Fehler #   : " & Str$(Err)
    Print #ErrFnum, "   VB Fehler     : " & Error$(Err)
    Print #ErrFnum, "   Oracle Fehler : " & gDatabase.LastServerErrText
    Print #ErrFnum, "   Datei #       : " & filename
    Print #ErrFnum, "   Arbeitsplan # : " & aplnr
    Print #ErrFnum, "    APLIndex #   : " & aplindex
    Print #ErrFnum, "     FHMNummer   : " & fhmnummer
    Print #ErrFnum, "     FHMTyp      : " & fhmtyp
    Print #ErrFnum, "   Fehlerhafte Zl: " & theLine
    Print #ErrFnum, ""
    Close #ErrFnum
    gDatabase.LastServerErrReset
    fhmlesen = 1
    Exit Function
End Function

'=========================================================
' Prueft, ob die uebergeben NC-Programmnummer im System
' existiert
'=========================================================
Function ncnrexists(ncnr As String) As Integer
    Dim theSet As Object

    Set theSet = gDatabase.DbCreateDynaset("select ncnr from zncnr where ncnr = '" &
            ncnr & "'", 0&)
    ncnrexists = Not theSet.EOF()

End Function
```

```
'=========================================================
' Liest die Bearbeitungsplatzzuordnungen zu einer Position
'=========================================================
Function platzlesen(filename As String, aplnr As String, herstid As String, aplindex As
           Long, FileNum As Integer) As Integer

    Dim theLine As String
    Dim zuord As Integer
    Dim ErrFnum As Integer
    Dim theTestSet As Object
    Dim platznr As String

    platzlesen = -1
    On Error GoTo fehlerbehandlung4
    theLine = Trim(theLine)

    While (Left$(theLine, 8) <> "PLATZEND") And (Not EOF(FileNum))

        Line Input #FileNum, theLine

        Select Case Left$(theLine, 15)
            Case "PLATZ........."
                platznr = Left$(Trim(right$(theLine, Len(theLine) - 16)), 16)
                If (gDatabase.DbExecuteSQL("insert into zavo_bea_1 (beapl_platznr,
                                avo_apl_arbeitspla, avo_apl_herst_id, avo_apl_bak,
                                avo_apl_index) select platznr, '" & aplnr & "', '"
                                & herstid & "', 0, " & aplindex & " from
                                zbearbeitungsplaetze where tritonname = '" &
                                platznr & "'") <= 0) Then
                    ErrFnum = FreeFile
                    Open globallogfile For Append Access Write As #ErrFnum
                    Print #ErrFnum, "***WARNING*** " & Date$ & " " & Time$
                    Print #ErrFnum, "   Ungueltige Platzangabe : " & platznr
                    Print #ErrFnum, "   Datei #       : " & filename
                    Print #ErrFnum, "   Arbeitsplan # : " & aplnr
                    Print #ErrFnum, "    APLIndex #   : " & aplindex
                    Print #ErrFnum, "   Fehlerhafte Zl: " & theLine
                    Print #ErrFnum, ""
                    Close #ErrFnum
                    platzlesen = 1
                End If
        End Select
    Wend
    Exit Function

fehlerbehandlung4:

    ErrFnum = FreeFile
    Open globallogfile For Append Access Write As #ErrFnum
    Print #ErrFnum, "***WARNING*** " & Date$ & " " & Time$
    Print #ErrFnum, "   VB Fehler #   : " & Str$(Err)
    Print #ErrFnum, "   VB Fehler     : " & Error$(Err)
    Print #ErrFnum, "   Oracle Fehler : " & gDatabase.LastServerErrText
    Print #ErrFnum, "   Datei #       : " & filename
    Print #ErrFnum, "   Arbeitsplan # : " & aplnr
    Print #ErrFnum, "    APLIndex #   : " & aplindex
    Print #ErrFnum, "     Platznr     : " & platznr
    Print #ErrFnum, "   Fehlerhafte Zl: " & theLine
    Print #ErrFnum, ""
    Close #ErrFnum
    gDatabase.LastServerErrReset
    platzlesen = 1
    Exit Function
End Function

'=========================================================
' Einlesen des Arbeitsplankopfes aus der Datei
'=========================================================
Function ReadAPL(aplnr As String, prefix As String, FileNum As Integer, aupnr As String,
           checkauftrag As Integer) As Integer

    ReadAPL = False
    Dim I As Integer
    Dim theSet As Object
    Dim theStatSet As Object
    Dim theAktSet As Object
    Dim herstid As String
```

```
Dim gesperrt As Integer
Dim theLine As String
Dim ErrFnum As Integer
Dim theDate, theTime As String
Dim theErstellDate, theErstellTime As String
Dim theTerminDate, theTerminTime As String
Dim theLieferDate, theLieferTime As String
Dim srcname, destname As String
Dim backup As Integer

On Error GoTo fehlerbehandlung

Set theSet = gDatabase.DbCreateDynaset("select ftgauftrag from zauftraege where
        apl_arbeitspla = '" & aplnr & "'", 0&)
If (Not theSet.EOF()) Then
    'Fertigungsauftrag auf Arbeitsplan freigegeben !
    ErrFnum = FreeFile
    Open globallogfile For Append Access Write As #ErrFnum
    Print #ErrFnum, "***ERROR*** " & Date$ & " " & Time$
    Print #ErrFnum, "   DateiName     : " & prefix
    Print #ErrFnum, "   Bemerkung     : Kann Arbeitsplan " & aplnr & " nicht
                einlesen, Fertigungsauftrag existiert !"
    Close #ErrFnum
    Exit Function
End If

gesperrt = 1
backup = False
'pruefen ob vorhanden
Set theSet = gDatabase.DbCreateDynaset("select gesperrt from zapls where arbeitspla
        = '" & aplnr & "' and bak = 0", 0&)
If (Not theSet.EOF()) Then
    'vorhanden
    gesperrt = theSet.Fields("gesperrt").Value
    backup = True
    'pruefen ob backup vorhanden
    Set theSet = gDatabase.DbCreateDynaset("select arbeitspla from zapls where
                arbeitspla = '" & aplnr & "' and bak = 1", 0&)
    If (Not theSet.EOF()) Then
        'backup vorhanden
            gDatabase.DbExecuteSQL ("delete from zavo_bea_1 where avo_apl_bak = 1
                and avo_apl_arbeitspla = '" & aplnr & "'")
             gDatabase.DbExecuteSQL ("delete from zvibs where avo_apl_bak = 1 and
                avo_apl_arbeitspla = '" & aplnr & "'")
        gDatabase.DbExecuteSQL ("delete from zplanungstermine where avo_apl_bak = 1
                and avo_apl_arbeitspla = '" & aplnr & "'")
        gDatabase.DbExecuteSQL ("delete from zavo_bearb where avo_apl_bak = 1 and
                avo_apl_arbeitspla = '" & aplnr & "'")
        gDatabase.DbExecuteSQL ("delete from zavo where apl_bak = 1 and
                apl_arbeitspla = '" & aplnr & "'")
        gDatabase.DbExecuteSQL ("delete from zauftraege where apl_bak = 1 and
                apl_arbeitspla = '" & aplnr & "'")
        gDatabase.DbExecuteSQL ("delete from zapls where bak = 1 and arbeitspla = '"
                & aplnr & "'")
    End If
    gDatabase.DbExecuteSQL ("insert into zapls (arbeitspla, herst_id, bak,
           ersteller, erstelldz, gesperrt, kommentar) select arbeitspla, herst_id,
           1, ersteller, erstelldz, gesperrt, kommentar from zapls where arbeitspla
           = '" & aplnr & "' and bak = 0")
    gDatabase.DbExecuteSQL ("insert into zauftraege (ftgauftrag, status, ersteller,
           erstelldz, stueck, prioritaet, planfreig, fertfreig, bezeichng, projekt,
           herst_id, stueck_ist, prio_ist, plfrei_ist, fefrei_ist, mandant,
           kommentar, lieferung, apl_arbeitspla, apl_herst_id, apl_bak, ftgstufe,
           ablieferort) select ftgauftrag, status, ersteller, erstelldz, stueck,
           prioritaet, planfreig, fertfreig, bezeichng, projekt, herst_id,
           stueck_ist, prio_ist, plfrei_ist, fefrei_ist, mandant, kommentar,
           lieferung, apl_arbeitspla, apl_herst_id, 1, ftgstufe, ablieferort from
           zauftraege where apl_arbeitspla = '" & aplnr & "' and apl_bak = 0")
    gDatabase.DbExecuteSQL ("insert into zavo (apl_arbeitspla, apl_herst_id,
           apl_bak, apl_index, avg_nr, stueckzeit, stueck, gesperrt, messfreq,
           kommentar, automatik, ncnr_ncnr, avgtyp_avgtyp) select apl_arbeitspla,
           apl_herst_id, 1, apl_index, avg_nr, stueckzeit, stueck, gesperrt,
           messfreq, kommentar, automatik, ncnr_ncnr, avgtyp_avgtyp from zavo where
           apl_arbeitspla = '" & aplnr & "' and apl_bak = 0")
    gDatabase.DbExecuteSQL ("insert into zavo_bearb (avo_apl_herst_id,avo_apl_index,
           avo_apl_arbeitspla, avo_apl_bak, bearb_typ_typ, bearb_nr) select
           avo_apl_herst_id, avo_apl_index, avo_apl_arbeitspla, 1, bearb_typ_typ,
```

```
            bearb_nr from zavo_bearb where avo_apl_arbeitspla = '" & aplnr & "' and
            avo_apl_bak = 0")
        gDatabase.DbExecuteSQL ("insert into zvibs (auftrag_ftgauftrag, status,
            avo_apl_herst_id, avo_apl_index, avo_apl_arbeitspla, avo_apl_bak,
            datum) select auftrag_ftgauftrag, status, avo_apl_herst_id,
            avo_apl_index, avo_apl_arbeitspla, 1, datum from zvibs where
            avo_apl_arbeitspla = '" & aplnr & "' and avo_apl_bak = 0")
        gDatabase.DbExecuteSQL ("insert into zplanungstermine (auftrag_ftgauftrag,
            avo_apl_herst_id, avo_apl_index, avo_apl_arbeitspla, avo_apl_bak, anfang,
            ende_frueh) select auftrag_ftgauftrag, avo_apl_herst_id, avo_apl_index,
            avo_apl_arbeitspla, 1, anfang, ende_frueh from zplanungstermine where
            avo_apl_arbeitspla = '" & aplnr & "' and avo_apl_bak = 0")
        gDatabase.DbExecuteSQL ("insert into zavo_bea_1 (avo_apl_herst_id,
            avo_apl_index, avo_apl_arbeitspla, avo_apl_bak, beapl_platznr)
            select avo_apl_herst_id, avo_apl_index, avo_apl_arbeitspla, 1,
            beapl_platznr from zavo_bea_1 where avo_apl_arbeitspla = '" &
            aplnr & "' and avo_apl_bak = 0")
        gDatabase.DbExecuteSQL ("delete from zavo_bea_1 where avo_apl_bak = 0 and
            avo_apl_arbeitspla = '" & aplnr & "'")
        gDatabase.DbExecuteSQL ("delete from zvibs where avo_apl_bak = 0 and
            avo_apl_arbeitspla = '" & aplnr & "'")
        gDatabase.DbExecuteSQL ("delete from zplanungstermine where avo_apl_bak = 0 and
            avo_apl_arbeitspla = '" & aplnr & "'")
        gDatabase.DbExecuteSQL ("delete from zavo_bearb where avo_apl_bak = 0 and
            avo_apl_arbeitspla = '" & aplnr & "'")
        gDatabase.DbExecuteSQL ("delete from zavo where apl_bak = 0 and apl_arbeitspla =
            '" & aplnr & "'")
        gDatabase.DbExecuteSQL ("delete from zauftraege where apl_bak = 0 and
            apl_arbeitspla = '" & aplnr & "'")
        gDatabase.DbExecuteSQL ("delete from zapls where bak = 0 and arbeitspla = '" &
            aplnr & "'")
    End If

    Set theSet = gDatabase.DbCreateDynaset("select arbeitspla, herst_id, bak, ersteller,
            erstelldz, gesperrt, kommentar from zapls", 0&)
    theSet.DbAddNew

    theSet.Fields("arbeitspla").Value = Left$(aplnr, 13)
    theSet.Fields("bak").Value = 0
    theSet.Fields("gesperrt").Value = gesperrt
    While (StrComp(Left$(theLine, 8), "AVOBEGIN") <> 0) And (Not EOF(FileNum))
        Select Case Left$(theLine, 15)
            Case "HERST_ID......."
                theSet.Fields("herst_id").Value = Left$(Trim$(right$(theLine,
                            Len(theLine) - 16)), 16)
                    herstid = Left$(Trim$(right$(theLine, Len(theLine) - 16)), 16)
            Case "ERSTELLER......"
                theSet.Fields("ersteller").Value = Left$(Trim$(right$(theLine,
                            Len(theLine) - 16)), 15)
            Case "KOMMENTAR......"
                theSet.Fields("kommentar").Value = Left$(Trim$(right$(theLine,
                            Len(theLine) - 16)), 60)
            Case "ERSTELLDZ......"
                theLine = Trim$(right$(theLine, Len(theLine) - 16))
                theDate = Left$(theLine, InStr(theLine, ":") - 1)
                theTime = Trim(right$(theLine, Len(theLine) - InStr(theLine, ":")))
                theSet.Fields("erstelldz").Value = "01-Jan-80"
        End Select
        Line Input #FileNum, theLine
    Wend
    theSet.DbUpdate

    gDatabase.DbExecuteSQL ("update zapls set erstelldz = TO_DATE ('" & theDate & " " &
            theTime & "', 'YYYY.MM.DD HH24:MI:SS') where arbeitspla = '" & aplnr & "'
            and bak = 0")

    If (Not checkauftrag) Then
        'auftragskopf einlesen

        Set theSet = gDatabase.DbCreateDynaset("select * from zauftraege", 0&)
        Seek #FileNum, 1
        theSet.DbAddNew

        'default werte schreiben
        theSet.Fields("apl_bak").Value = 0
        theSet.Fields("status").Value = 0
        theSet.Fields("stueck_ist").Value = 0
```

```
    theSet.Fields("prio_ist").Value = 9
    theSet.Fields("prioritaet").Value = 9
    theSet.Fields("ftgauftrag").Value = aupnr
    theSet.Fields("fefrei_ist").Value = 1
    theSet.Fields("fertfreig").Value = 1
    theSet.Fields("plfrei_ist").Value = 1
    theSet.Fields("planfreig").Value = 1
    theSet.Fields("apl_arbeitspla").Value = aplnr

    'jetzt loopen bis der arbeitsplan kommt
    Line Input #FileNum, theLine
    While (Left$(theLine, 10) <> "ARBEITSPLA") And (Not EOF(FileNum))
        Select Case Left$(theLine, 15)
            Case "ERSTELLER......"
                theSet.Fields("ersteller").Value = LTrim(right$(theLine,
                                Len(theLine) - 16))
            Case "KOMMENTAR......"
                theSet.Fields("kommentar").Value = LTrim(right$(theLine,
                                Len(theLine) - 16))
            Case "STUECK........."
                theSet.Fields("stueck").Value = LTrim(right$(theLine,
                                Len(theLine) - 16))
            Case "PROJEKT........"
                theSet.Fields("projekt").Value = LTrim(right$(theLine,
                                Len(theLine) - 16))
            Case "BEZEICHNG......"
                theSet.Fields("bezeichng").Value = LTrim(right$(theLine,
                                Len(theLine) - 16))
            Case "HERST_ID......."
                theSet.Fields("herst_id").Value = LTrim(right$(theLine,
                                Len(theLine) - 16))
                theSet.Fields("apl_herst_id").Value = LTrim(right$(theLine,
                                Len(theLine) - 16))
                herstid = LTrim(right$(theLine, Len(theLine) - 16))
            Case "MANDANT........"
                theSet.Fields("mandant").Value = LTrim(right$(theLine,
                                Len(theLine) - 16))
            Case "ERSTELLDZ......"
                theLine = LTrim$(RTrim$(right$(theLine, Len(theLine) - 16)))
                theErstellDate = Left$(theLine, InStr(theLine, ":") - 1)
                theErstellTime = LTrim(right$(theLine, Len(theLine) -
                                InStr(theLine, ":")))
                theSet.Fields("erstelldz").Value = "01-Jan-80"
            Case "FTGSTUFE......."
                theSet.Fields("ftgstufe").Value = LTrim(right$(theLine,
                                Len(theLine) - 16))
            Case "ABLIEFERORT...."
                theSet.Fields("ablieferort").Value = LTrim(right$(theLine,
                         Len(theLine) - 16))
            Case "LIEFERUNG......"
                theLine = LTrim$(RTrim$(right$(theLine, Len(theLine) - 16)))
                theLieferDate = Left$(theLine, InStr(theLine, ":") - 1)
                theLieferTime =  LTrim(right$(theLine, Len(theLine) - InStr(theLine,
                                ":")))
                theSet.Fields("lieferung").Value = "01-Jan-80"
        End Select
        Line Input #FileNum, theLine
    Wend
    theSet.DbUpdate

    'jetzt die datumswerte einfuegen
    gDatabase.DbExecuteSQL ("update zauftraege set erstelldz = TO_DATE ('" &
                  theErstellDate & " " & theErstellTime & "', 'YYYY.MM.DD
                  HH24:MI:SS') where ftgauftrag = '" & aupnr & "'")
    gDatabase.DbExecuteSQL ("update zauftraege set lieferung = TO_DATE ('" &
                  theLieferDate & " " & theLieferTime & "', 'YYYY.MM.DD
                  HH24:MI:SS') where ftgauftrag = '" & aupnr & "'")

    Seek #FileNum, 1
    While (Left$(theLine, 8) <> "AVOBEGIN") And (Not EOF(FileNum))
        Line Input #FileNum, theLine
    Wend
End If

While (Not EOF(FileNum))
    'Alle Arbeitsvorgaenge lesen
    If (Not avolesen(prefix, aplnr, herstid, FileNum, aupnr, Not checkauftrag)) Then
```

```
            GoTo fehlerbehandlung
        End If

        'weiter lesen bis dateiende oder naechster avo
        If (Not EOF(FileNum)) Then
            Do
                Line Input #FileNum, theLine
            Loop While (StrComp(Left$(theLine, 8), "AVOBEGIN") <> 0) And (Not
               EOF(FileNum))
        End If
    Wend

    If (backup) Then
        If (Not FHMAbgleich(aplnr, herstid)) Then
            GoTo fehlerbehandlung
        End If
    End If

    ErrFnum = FreeFile
    Open globallogfile For Append Access Write As #ErrFnum
    Print #ErrFnum, "***SUCCESS*** " & Date$ & " " & Time$
    Print #ErrFnum, "   Dateien #     : " & prefix
    Print #ErrFnum, ""
    Close #ErrFnum
    ReadAPL = True
    Exit Function

fehlerbehandlung:

    Close
    srcname = globalspfad & prefix
    destname = globalfehlerpfad & prefix
    FileCopy srcname, destname

    ErrFnum = FreeFile
    Open globallogfile For Append Access Write As #ErrFnum
    Print #ErrFnum, "***ERROR*** " & Date$ & " " & Time$
    Print #ErrFnum, "   VB Fehler #   : " & Str$(Err)
    Print #ErrFnum, "   VB Fehler     : " & Error$(Err)
    Print #ErrFnum, "   Oracle Fehler : " & gDatabase.LastServerErrText
    Print #ErrFnum, "   Datei #       : " & prefix
    Print #ErrFnum, "   Arbeitsplan # : " & aplnr
    Print #ErrFnum, "   Fehlerhafte Zl: " & theLine
    Print #ErrFnum, ""
    Close #ErrFnum
    gDatabase.LastServerErrReset
    ReadAPL = False
    Exit Function
End Function

'=============================================================
' Liest einen Termin zu einer Arbeitsplanposition
'=============================================================
Function terminlesen(filename As String, aplnr As String, herstid As String, aplindex As
            Long, aupnr As String, FileNum As Integer) As Integer

    Dim fhmtyp As String
    Dim ErrFnum As Integer
    Dim theLine, theDate, theTime As String
    Dim stmtxt As String

    terminlesen = 0
    On Error GoTo fehlerbehandlung5

    theLine = LTrim(RTrim(theLine))

    stmtxt = "insert into zvibs (auftrag_ftgauftrag, status, avo_apl_herst_id,
            avo_apl_index, avo_apl_arbeitspla, avo_apl_bak, datum) values "
    stmtxt = stmtxt & " ('" & aupnr & "', 0, '" & herstid & "', " & aplindex & ", '" &
            aplnr & "', 0, SYSDATE)"
    gDatabase.DbExecuteSQL (stmtxt)

    stmtxt = "insert into zplanungstermine (auftrag_ftgauftrag, avo_apl_herst_id,
            avo_apl_index, avo_apl_arbeitspla, avo_apl_bak, anfang, ende_frueh) values "
    stmtxt = stmtxt & " ('" & aupnr & "', '" & herstid & "', " & aplindex & ", '" &
            aplnr & "', 0, '01-JAN-80', '01-JAN-80')"
    gDatabase.DbExecuteSQL (stmtxt)
```

```
    While (Left$(theLine, 9) <> "TERMINEND") And (Not EOF(FileNum))
        Line Input #FileNum, theLine

        Select Case Left$(theLine, 15)
            Case "ANFANG........."
                theLine = Trim$(right$(theLine, Len(theLine) - 16))
                theDate = Left$(theLine, InStr(theLine, ":") - 1)
                theTime = Trim(right$(theLine, Len(theLine) - InStr(theLine, ":")))
                gDatabase.DbExecuteSQL ("update zplanungstermine set anfang = TO_DATE
                            ('" & theDate & " " & theTime & "', 'YYYY.MM.DD
                            HH24:MI:SS') where auftrag_ftgauftrag = '" & aupnr & "'
                            and avo_apl_herst_id = '" & herstid & „' and avo_apl_index
                            = " & aplindex & " and avo_apl_arbeitspla = '" & aplnr
                            & "' and avo_apl_bak = 0")
            Case "ENDEFRUEH......"
                theLine = Trim$(right$(theLine, Len(theLine) - 16))
                theDate = Left$(theLine, InStr(theLine, „:") - 1)
                theTime = Trim(right$(theLine, Len(theLine) - InStr(theLine, ":")))
                gDatabase.DbExecuteSQL ("update zplanungstermine set ende_frueh =
                            TO_DATE ('" & theDate & " " & theTime &
                             "', 'YYYY.MM.DD HH24:MI:SS') where auftrag_ftgauftrag = '"
                            & aupnr & "' and avo_apl_herst_id = '" & herstid &
                            "' and avo_apl_index = " & aplindex & " and
                            avo_apl_arbeitspla = '" & aplnr & "' and avo_apl_bak = 0")
        End Select
    Wend

    terminlesen = -1
    Exit Function

fehlerbehandlung5:

    ErrFnum = FreeFile
    Open globallogfile For Append Access Write As #ErrFnum
    Print #ErrFnum, "***WARNING*** " & Date$ & " " & Time$
    Print #ErrFnum, "   VB Fehler #   : " & Str$(Err)
    Print #ErrFnum, "   VB Fehler     : " & Error$(Err)
    Print #ErrFnum, "   Oracle Fehler : " & gDatabase.LastServerErrText
    Print #ErrFnum, "   Datei #       : " & filename
    Print #ErrFnum, "   Arbeitsplan # : " & aplnr
    Print #ErrFnum, "    APLIndex #   : " & aplindex
    Print #ErrFnum, "   Fehlerhafte Zl: " & theLine
    Print #ErrFnum, ""
    Close #ErrFnum
    gDatabase.LastServerErrReset
    terminlesen = 1
    Exit Function
End Function
```

Basic-Programm zur Fertigungshilfsmittelzuordnung

```
'=========================================================
' (C) ABB Kraftwerke AG
'
' Autor : Joerg Evermann
'         erstellt im Rahmen der Diplomhausarbeit,
'         Oktober 1996
'
'=========================================================
'=========================================================
' Allgemeine Deklarationen
'=========================================================
Option Explicit
Dim avogesperrt_dirty As Integer
Dim herst_id, apl_index, arbeitspla, theNr, theTyp As String
Private Declare Function WinExec Lib "kernel" (ByVal CmdLine As String, ByVal CmdShow As
            Integer) As Integer

'=========================================================
' Objekt : avot
' Event  : KeyPress
'
```

```
' Registriert die Veraenderung des Eingabefeldes
'=========================================================
Private Sub avot_KeyPress(Index As Integer, KeyAscii As Integer)
    avogesperrt_dirty = True
End Sub

'=========================================================
' Objekt : cmdanzeigen
' Event  : Click
'
' Registriert das Clicken auf den Anzeigeknopf
'=========================================================
Private Sub cmdAnzeigen_Click()

    Dim I, TotalSize, fnum, RetVal, NumChunks, RemChunkSize, ChunkSize, CurSize As Long
    Dim beadatset As Object
    Dim theVer As Single
    Dim aktver As Double
    Dim StmtText, stmtxt, theLine, herst_id, arbeitspla, apl_index, theNr, Fname,
            theEditor, theTyp As String
    Dim CurChunk, theCallStr As String

    theLine = Liste1.FullPath(Liste1.ListIndex)

    herst_id = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, "\") - 1)))
    theLine = Right$(theLine, Len(theLine) - InStr(theLine, "\"))
    arbeitspla = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, "\") - 1)))
    theLine = Right$(theLine, Len(theLine) - InStr(theLine, "\"))
    apl_index = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, "\") - 1)))
    theLine = Right$(theLine, Len(theLine) - InStr(theLine, "\"))
    theNr = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, ".") - 1)))
    theLine = Right$(theLine, Len(theLine) - InStr(theLine, "."))
    theTyp = LTrim(RTrim(theLine))

    Select Case theTyp
        Case "NC"
            stmtxt = "select ver from zncnrver where aktver = 1 and ncnr_ncnr = '" &
               theNr & "'"
            Set beadatset = gDatabase.DbCreateDynaset(stmtxt, 0&)
            beadatset.DbMoveFirst
            aktver = beadatset.Fields("ver").Value

            stmtxt = "select * from zbeadats where beaart_dtk = 'NC' and
                      ncnrver_ncnr_ncnr = '" & theNr & "' and ncnrver_ver = " & aktver
            Set beadatset = gDatabase.DbCreateDynaset(stmtxt, 0&)
            If (Not beadatset.EOF()) Then
                beadatset.DbMoveFirst
                Fname = "c:\" & theNr & ".nc"
                ChunkSize = 1024& * 16&
                fnum = FreeFile ' Get free file number.
                Open Fname For Binary As #fnum
                I = 0
                Do
                    CurChunk = Beadatset.Fields("datum").DbGetChunk(I * ChunkSize + 1,
                                        ChunkSize)
                    CurSize = Len(CurChunk)
                    Put #fnum, , CurChunk
                    I = I + 1
                Loop Until CurSize < ChunkSize
                Close fnum

                stmtxt = "select editor from zbeaarts where dtk = 'NC'"
                Set beadatset = gDatabase.DbCreateDynaset(stmtxt, 0&)
                beadatset.DbMoveFirst
                theEditor = beadatset.Fields("editor").Value
                theCallStr = theEditor & " " & Fname
                RetVal = WinExec(theCallStr, 1)
            End If
        Case Else
            stmtxt = "select * from zbeaver where bearb_nr = '" & theNr & "' and aktver
                      = 1 and bearb_typ_typ = '" & theTyp & "'"
            Set beadatset = gDatabase.DbCreateDynaset(stmtxt, 0&)
            If (Not beadatset.EOF()) Then
                beadatset.DbMoveFirst
                Fname = "c:\" & theNr & "." & theTyp
                ChunkSize = 1024& * 16&
                fnum = FreeFile ' Get free file number.
```

```
                Open Fname For Binary As #fnum
                I = 0
                Do
                    CurChunk = beadatset.Fields("daten").DbGetChunk(I * ChunkSize + 1,
                                        ChunkSize)
                    CurSize = Len(CurChunk)
                    Put #fnum, , CurChunk
                    I = I + 1
                Loop Until CurSize < ChunkSize
                Close fnum
                stmtxt = "select editor from ztyp where typ = '" & theTyp & "'"
                Set beadatset = gDatabase.DbCreateDynaset(stmtxt, 0&)
                beadatset.DbMoveFirst

                theEditor = beadatset.Fields("editor").Value
                theCallStr = theEditor & " " & Fname
                RetVal = WinExec(theCallStr, 1)
            End If
    End Select
End Sub

'=========================================================
' Objekt : cmdentfernen
' Event  : Click
'
' Registriert das Clicken auf den Entfernenknopf
'=========================================================
Private Sub cmdentfernen_Click()

    Dim I As Long
    Dim theSet As Object
    Dim stmtxt As String
    Dim theLine, theTyp, theNr, herst_id, arbeitspla, apl_index As String
    On Error GoTo cmdentfernen_Click_ERROR
    theLine = Liste1.FullPath(Liste1.ListIndex)
    herst_id = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, "\") - 1)))
    theLine = Right$(theLine, Len(theLine) - InStr(theLine, "\"))
    arbeitspla = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, "\") - 1)))
    theLine = Right$(theLine, Len(theLine) - InStr(theLine, "\"))
    apl_index = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, "\") - 1)))
    theLine = Right$(theLine, Len(theLine) - InStr(theLine, "\"))
    theNr = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, ".") - 1)))
    theLine = Right$(theLine, Len(theLine) - InStr(theLine, "."))
    theTyp = theLine

    Select Case theTyp
        Case "NC"
            stmtxt = "update zavo set ncnr_ncnr = NULL where apl_arbeitspla = '" &
                     arbeitspla & "' and apl_herst_id = '" & herst_id & "' and
                     apl_index = '" & apl_index & "' and apl_bak = 0"
            gDatabase.DbExecuteSQL (stmtxt)
            Liste1.RemoveItem (Liste1.ListIndex)
            Liste1.ListIndex = -1
        Case Else
            stmtxt = "delete from zavo_bearb where avo_apl_index = '" & apl_index & "'
                     and avo_apl_arbeitspla = '" & arbeitspla & "' and avo_apl_herst_id
                     = '" & herst_id & "' and avo_apl_bak = 0 and bearb_typ_typ = '" &
                     theTyp & "' and bearb_nr = '" & theNr & "'"
            gDatabase.DbExecuteSQL (stmtxt)
            Liste1.RemoveItem (Liste1.ListIndex)
            Liste1.ListIndex = -1
    End Select
Exit Sub
cmdentfernen_Click_ERROR:

    Screen.MousePointer = 0
    If gDatabase.LastServerErr <> 0 Then
        MsgBox gDatabase.LastServerErrText, MB_OK, "Fehler"
    Else 'Must be some non-Oracle error
        MsgBox "VB:" & Err & " " & Error(Err), MB_OK, "Fehler"
    End If
Exit Sub
End Sub

'=========================================================
' Objekt : cmdhinzufuegen
' Event  : Click
```

```
'
' Registriert das Clicken auf den Hinzufuegenknopf
'=========================================================
Private Sub cmdhinzufuegen_Click()

    Dim I, j As Long
    Dim theSet As Object
    Dim stmtxt As String
    Dim theLine, theTyp, theNr, herst_id, arbeitspla, apl_index As String

    If (Liste2.ListIndex <> -1) Then
        If (Liste2.Indent(Liste2.ListIndex) = 2) Then
            I = Liste2.ListIndex
            theLine = Liste2.FullPath(I)
            theNr = LTrim(RTrim(Mid$(theLine, InStr(theLine, "\") + 1, 16)))
            theTyp = LTrim(RTrim(Mid$(theLine, 1, 3)))
            Select Case theTyp
                Case "NC\"
                    theLine = Liste1.FullPath(Liste1.ListIndex)
                    herst_id = LTrim(RTrim(Mid$(theLine, 1,InStr(theLine, "\") - 1)))
                    theLine = Right$(theLine, Len(theLine)-InStr(theLine, "\"))
                    arbeitspla = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, "\") - 1)))
                    theLine = Right$(theLine, Len(theLine)-InStr(theLine, "\"))
                    apl_index = theLine

                    stmtxt = "update zavo set ncnr_ncnr = '" & theNr & "' where
                                    apl_arbeitspla = '" & arbeitspla & "' and
                                    apl_herst_id = '" & herst_id & "' and apl_index =
                                    '" & apl_index & "' and apl_bak = 0"
                    gDatabase.DbExecuteSQL (stmtxt)
                    Liste2.ListIndex = -1
                    j = 1
                    If (Liste1.ListIndex + j < Liste1.ListCount) Then
                        While (Liste1.Indent(Liste1.ListIndex +j) > Liste1.Indent(
                                                Liste1.ListIndex))
                            If (StrComp(Right$(Liste1.List(Liste1.ListIndex + j), 2),
                                        "NC") = 0) Then
                                Liste1.RemoveItem Liste1.ListIndex + j
                                Liste1.AddItem theNr & "."& "NC", Liste1.ListIndex+j
                                Liste1.Indent(Liste1.ListIndex + j) = 4
                                Liste1.PictureType(Liste1.ListIndex + j) = 2
                                Liste1.Expand(Liste1.ListIndex) = True
                                Exit Sub
                            End If
                            j = j + 1
                            If (Liste1.ListIndex + j >= Liste1.ListCount) Then
                                Liste1.AddItem theNr&"." & "NC", Liste1.ListIndex+1
                                Liste1.Indent(Liste1.ListIndex+1) = 4
                              Liste1.PictureType(Liste1.ListIndex + 1)=2
                              Liste1.Expand(Liste1.ListIndex) = True
                              Exit Sub
                            End If
                        Wend
                    End If
                    Liste1.AddItem theNr & "." & "NC",Liste1.ListIndex + 1
                    Liste1.Indent(Liste1.ListIndex + 1) = 4
                    Liste1.PictureType(Liste1.ListIndex + 1) = 2
                    Liste1.Expand(Liste1.ListIndex) = True
                    Exit Sub
                Case Else
                    theLine = Liste1.FullPath(Liste1.ListIndex)
                    herst_id = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, "\") - 1)))
                    theLine = Right$(theLine, Len(theLine)- InStr(theLine, "\"))
                    arbeitspla = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, "\") - 1)))
                    theLine = Right$(theLine, Len(theLine)- InStr(theLine, "\"))
                    apl_index = theLine

                    On Error GoTo fehler
                    stmtxt = "insert into zavo_bearb (avo_apl_index,avo_apl_arbeitspla,
                                    avo_apl_herst_id, avo_apl_bak, bearb_typ_typ,
                                    bearb_nr) values ('" & apl_index & "', '" &
                                    arbeitspla & "', '" & herst_id & "', 0, '" & theTyp
                                    & "', '" & theNr & "')"
                    gDatabase.DbExecuteSQL (stmtxt)
                    On Error GoTo 0
                    Liste2.ListIndex = -1
                    Liste1.AddItem theNr & "." & theTyp, Liste1.ListIndex + 1
```

```
                    Liste1.Indent(Liste1.ListIndex + 1) = 4
                    Liste1.PictureType(Liste1.ListIndex + 1) = 2
                    Liste1.Expand(Liste1.ListIndex) = True
            End Select
        End If
    End If

fehler:
    Exit Sub
End Sub

'=========================================================
' Objekt : Form FHMZ
' Event  : Load
'
' Wird ausgefuehrt, wenn das Formular geoeffnet wird
'=========================================================
Private Sub Form_Load()
    Dim theTeilSet As Object
    Dim theAPLSet As Object
    Dim theAVOSet As Object
    Dim theBeadatSet As Object

    Dim I As Long

    For I = 0 To 1
        apl(I).Visible = False
        apl(I).BackColor = &HC0C0C0
        aplt(I).Visible = False
    Next I
    For I = 0 To 7
        avo(I).Visible = False
        avo(I).BackColor = &HC0C0C0
        avot(I).Visible = False
    Next I
    cmdanzeigen.Visible = False

    Set theTeilSet = gDatabase.DbCreateDynaset("select distinct herst_id from zapls
            order by herst_id desc", 0&)
    I = 1
    While (Not theTeilSet.EOF())
        Liste1.AddItem theTeilSet.Fields("herst_id").Value,I
        theTeilSet.DbMoveNext
        Liste1.Indent(I) = 1
    Wend

    cmdhinzufuegen.Enabled = False
    cmdentfernen.Enabled = False

    avogesperrt_dirty = False
End Sub

'=========================================================
' Objekt : Form FHMZ
' Event  : QueryUnload
'
' Wird ausgefuehrt, bevor das Formular geschlossen wird
'=========================================================
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    If (avogesperrt_dirty = True) Then
        If (StrComp(avot(5).Text, "1") = 0 Or StrComp(avot(5), "0") = 0) Then
            gDatabase.DbExecuteSQL ("update zavo set gesperrt = '" & avot(5).Text & "'
                    where apl_index = '" & apl_index & "' and apl_arbeitspla = '" &
                    arbeitspla & "' and apl_herst_id = '" & herst_id & "' and apl_bak
                    = 0")
        End If
        avogesperrt_dirty = False
    End If
End Sub

'=========================================================
' Objekt : Liste1
' Event  : Click
'
' Registriert das Clicken auf einen Eintrag der Liste
'=========================================================
Private Sub Liste1_Click()
```

```
Dim theBeadatSet As Object
Dim theTRIPLSet As Object
Dim theSet As Object
Dim stmtxt As String
Dim triplatz As String
Dim ListIndex As Integer
Dim theLine As String
Dim I As Long

Screen.MousePointer = 11
ListIndex = Liste1.ListIndex

If (ListIndex <> -1) Then
    If (avogesperrt_dirty = True) Then
        If (StrComp(avot(5).Text, "1") = 0 Or StrComp(avot(5), "0") = 0) Then
            gDatabase.DbExecuteSQL ("update zavo set gesperrt = '" & avot(5).Text &
                        "' where apl_index = '" & apl_index & "' and apl_arbeitspla
                        = '" & arbeitspla & "' and apl_herst_id = '" & herst_id &"'
                        and apl_bak = 0")
        End If
        avogesperrt_dirty = False
    End If

    Select Case (Liste1.Indent(ListIndex))
        Case 1
            'Ein Herstellteil ist ausgewaehlt
            Liste2.Clear
            For I = 0 To 1
                apl(I).Visible = False
                aplt(I).Visible = False
            Next I
            For I = 0 To 7
                avo(I).Visible = False
                avot(I).Visible = False
            Next I
            cmdanzeigen.Visible = False
            cmdhinzufuegen.Enabled = False
            cmdentfernen.Enabled = False

            Frame3D1.Caption = "Herstell ID"
        Case 2
            'Ein Arbeitsplan ist ausgewaehlt
            Liste2.Clear
            For I = 0 To 1
                apl(I).Visible = True
                aplt(I).Visible = True
            Next I
            For I = 0 To 7
                avo(I).Visible = False
                avot(I).Visible = False
            Next I
            cmdanzeigen.Visible = False

            theLine = Liste1.FullPath(Liste1.ListIndex)
            herst_id = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, "\") - 1)))
            theLine = Right$(theLine, Len(theLine) - InStr(theLine, "\"))
            arbeitspla = theLine

            Set theSet = gDatabase.DbCreateDynaset("select ersteller, kommentar from
                        zapls where bak = 0 and herst_id = '" & herst_id & "' and
                        arbeitspla = '" & arbeitspla & "'", 0&)
            theSet.DbMoveFirst

            For I = 0 To 1
                aplt(I).Text = theSet.Fields(I).Value
            Next I

            cmdhinzufuegen.Enabled = False
            cmdentfernen.Enabled = False

            Frame3D1.Caption = "Arbeitsplan"
        Case 3
            'Ein Arbeitsvorgang ist ausgewaehlt
            'Jetzt die Liste aller moeglichen NC Programme
            'und Bearbeitungsdateine aktualisieren
```

```
                cmdhinzufuegen.Enabled = True
                cmdentfernen.Enabled = False
                For I = 0 To 1
                    apl(I).Visible = False
                    aplt(I).Visible = False
                Next I
                For I = 0 To 7
                    avo(I).Visible = True
                    avot(I).Visible = True
                Next I
                cmdanzeigen.Visible = False

                I = 0

                Liste2.Clear
                Liste2.AddItem "NC", I
                Liste2.Indent(I) = 1

                Set theSet = gDatabase.DbCreateDynaset("select typ from ztyp", 0&)
                theSet.DbMoveFirst
                While (Not theSet.EOF)
                    I = I + 1
                    Liste2.AddItem theSet.Fields("typ").Value
                    Liste2.Indent(I) = 1
                    theSet.DbMoveNext
                Wend

                theLine = Liste1.FullPath(ListIndex)
                herst_id = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, "\") - 1)))
                theLine = Right$(theLine, Len(theLine) - InStr(theLine, "\"))
                arbeitspla = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, "\") - 1)))
                theLine = Right$(theLine, Len(theLine) - InStr(theLine, "\"))
                apl_index = theLine

                stmtxt = "select avg_nr, stueck, stueckzeit, kommentar, avgtyp_avgtyp,
                             gesperrt, automatik, messfreq from zavo where "
                stmtxt = stmtxt & " apl_arbeitspla = '" & arbeitspla & "' and apl_index
                             = '" & apl_index & "' and apl_herst_id = '" &
                             herst_id & "' and apl_bak = 0"
                Set theSet = gDatabase.DbCreateDynaset(stmtxt,0&)
                theSet.DbMoveFirst

                For I = 0 To 7
                    If Not (IsNull(theSet.Fields(I))) Then
                        avot(I).Text = theSet.Fields(I).Value
                    Else
                        avot(I).Text = ""
                    End If
                Next I

                Frame3D1.Caption = "Arbeitsvorgang"
            Case 4
                'Ein Bearbeitungsdatum ist ausgewaehlt
                Liste2.Clear
                For I = 0 To 1
                    apl(I).Visible = False
                    aplt(I).Visible = False
                Next I
                For I = 0 To 7
                    avo(I).Visible = False
                    avot(I).Visible = False
                Next I

                cmdanzeigen.Visible = True

                cmdhinzufuegen.Enabled = False
                cmdentfernen.Enabled = True

                Frame3D1.Caption = "Fertigungshilfsmitteldaten"
        End Select

    End If

    Screen.MousePointer = 0

End Sub
```

```
'=========================================================
' Objekt : Liste1
' Event  : Collapse
'
' Aufforderung zum Einklappen der Untereintraege eines
' Listeneintrags
'=========================================================
Private Sub Liste1_Collapse(ListIndex As Integer)

    Dim I As Long
    Screen.MousePointer = 11
    While (Liste1.Indent(ListIndex + 1) > Liste1.Indent(ListIndex))
        Liste1.RemoveItem (ListIndex + 1)
        If (ListIndex + 1) = Liste1.ListCount Then
            Liste1.PictureType(ListIndex) = 0
            Liste2.Clear
            Screen.MousePointer = 0
            Exit Sub
        End If
    Wend
    Liste1.PictureType(ListIndex) = 0
    Liste2.Clear
    Screen.MousePointer = 0

End Sub

'=========================================================
' Objekt : Liste1
' Event  : DblClick
'
' Registriert das Doppelclicken auf einen Eintrag der Liste
'=========================================================
Private Sub Liste1_DblClick()
    Screen.MousePointer = 11
    If Liste1.Indent(Liste1.ListIndex) = 4 Then
        Call cmdentfernen_Click
    Else
        Liste1.Expand(Liste1.ListIndex) = Not Liste1.Expand(Liste1.ListIndex)
        Liste1_Click
    End If
    Screen.MousePointer = 0
End Sub

'=========================================================
' Objekt : Liste1
' Event  : Expand
'
' Fordert das Anzeigen der Untereintraege eines
' Listeneintrags
'=========================================================
Private Sub Liste1_Expand(ListIndex As Integer)

    Dim theAPLSet As Object
    Dim theAVOSet As Object
    Dim theBeadatSet As Object
    Dim stmtxt As String
    Dim theLine As String
    Dim herst_id, apl_index, arbeitspla As String
    Dim Index As Long
    Dim I As Long
    Index = ListIndex

    Screen.MousePointer = 11

    Select Case Liste1.Indent(ListIndex)
        Case 1
            'Hier soll eine HerstellID expandiert werden
            If Liste1.HasSubItems(ListIndex) = False Then
                'Die dinger muessen noch eingelesen werden
                stmtxt = "select distinct arbeitspla from zapls where bak = 0 and
                             herst_id = '" & Liste1.List(ListIndex) & "' order by
                             arbeitspla desc"
                Set theAPLSet = gDatabase.DbCreateDynaset(stmtxt, 0&)
                I = 1
                While (Not theAPLSet.EOF())
                    Liste1.AddItem theAPLSet.Fields("arbeitspla").Value, ListIndex + I
                    Liste1.Indent(ListIndex + I) = 2
```

```
            theAPLSet.DbMoveNext
        Wend

        If Liste1.HasSubItems(ListIndex) = False Then
            'wenn keine hinzugefuegt werden konnten
            Screen.MousePointer = 0
            Exit Sub
        Else
            'wir haben welche hinzugefuegt, also jetzt expandieren
            Liste1.Expand(Index) = True
        End If
    Else
        Liste1.PictureType(ListIndex) = 1
    End If
Case 2
    'Hier soll ein Arbeitsplan expandiert werden
    theLine = Liste1.FullPath(ListIndex)
    herst_id = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, "\") - 1)))
    theLine = Right$(theLine, Len(theLine) - InStr(theLine, "\"))
    arbeitspla = theLine

    If Liste1.HasSubItems(ListIndex) = False Then
        'Die dinger muessen noch eingelesen werden
        stmtxt = "select distinct apl_index from zavo where apl_bak = 0 and
                    apl_herst_id = '" & herst_id & "' and apl_arbeitspla
                    = '" & arbeitspla & "' order by apl_index desc"
        Set theAVOSet = gDatabase.DbCreateDynaset(stmtxt,  0&)
        I = 1
        While (Not theAVOSet.EOF())
            Liste1.AddItem theAVOSet.Fields("apl_index").Value, ListIndex + I
            Liste1.Indent(ListIndex + I) = 3
            theAVOSet.DbMoveNext
        Wend

        If Liste1.HasSubItems(ListIndex) = False Then
            'wenn keine hinzugefuegt werden konnten
            Screen.MousePointer = 0
            Exit Sub
        Else
            'wir haben welche hinzugefuegt, also jetzt expandieren
            Liste1.Expand(Index) = True
        End If
    Else
        Liste1.PictureType(ListIndex) = 1
    End If
Case 3
    'Hier soll ein Arbeitsvorgang expandiert werden
    theLine = Liste1.FullPath(ListIndex)
    herst_id = LTrim(RTrim(Mid$(theLine, 1, InStr(theLine, "\") - 1)))
    theLine = Right$(theLine, Len(theLine) - InStr(theLine, "\"))
    arbeitspla = LTrim(RTrim(Mid$(theLine, 1,InStr(theLine, "\") - 1)))
    theLine = Right$(theLine, Len(theLine) - InStr(theLine, "\"))
    apl_index = theLine

    If Liste1.HasSubItems(ListIndex) = False Then
        I = 1
        'Die dinger muessen noch eingelesen werden
        stmtxt = "select ncnr_ncnr from zavo where apl_bak = 0 and apl_herst_id
                    = '" & herst_id & "' and apl_arbeitspla = '" & arbeitspla &
                    "' and apl_index = '" & apl_index & "' order by ncnr_ncnr
                    desc"
        Set theBeadatSet = gDatabase.DbCreateDynaset(stmtxt,0&)
        If (Not IsNull(theBeadatSet.Fields("ncnr_ncnr").Value)) Then
            Liste1.AddItem theBeadatSet.Fields("ncnr_ncnr").Value & ".NC",
                            ListIndex + 1
            Liste1.Indent(ListIndex + 1) = 4
            Liste1.PictureType(ListIndex + 1) = 2
            I = 2
        End If

        stmtxt = "select distinct bearb_nr, bearb_typ_typ from zavo_bearb where
                    avo_apl_bak = 0 and avo_apl_herst_id = '" & herst_id & "'
                    and avo_apl_arbeitspla = '" & arbeitspla & "' and
                    avo_apl_index = '" & apl_index & "' order by bearb_typ_typ
                    desc, bearb_nr desc"
            Set theBeadatSet = gDatabase.DbCreateDynaset(stmtxt, 0&)
        While (Not theBeadatSet.EOF())
```

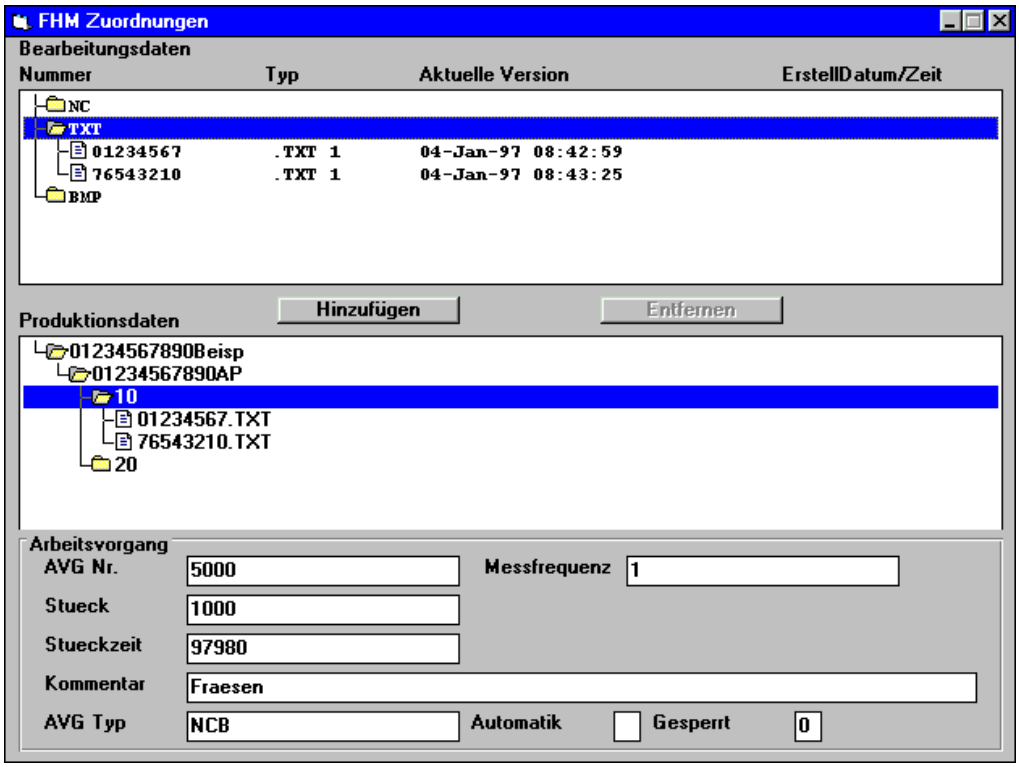
```
                    Liste1.AddItem theBeadatSet.Fields("bearb_nr").Value & "." &
                              theBeadatSet.Fields("bearb_typ_typ").Value,ListIndex+I
                    Liste1.Indent(ListIndex + I) = 4
                    Liste1.PictureType(ListIndex + I) = 2
                    theBeadatSet.DbMoveNext
                Wend

                If Liste1.HasSubItems(ListIndex) = False Then
                    'wenn keine hinzugefuegt werden konnten
                    Screen.MousePointer = 0
                    Exit Sub
                Else
                    'wir haben welche hinzugefuegt, also jetzt
                    'expandieren
                    Liste1.Expand(Index) = True
                End If
            Else
                Liste1.PictureType(ListIndex) = 1
            End If
    End Select

    Screen.MousePointer = 0

End Sub

'=========================================================
' Objekt : Liste1
' Event  : PictureClick
'
' Registriert das Clicken auf ein Icon in der Liste
'=========================================================
Private Sub Liste1_PictureClick(ListIndex As Integer)
    Screen.MousePointer = 11
    Liste1.ListIndex = ListIndex
    Liste1_Click
    Screen.MousePointer = 0
End Sub

'=========================================================
' Objekt : Liste1
' Event  : PictureDblClick
'
' Registriert das Doppelclicken auf ein Icon in der Liste
'=========================================================
Private Sub Liste1_PictureDblClick(ListIndex As Integer)
    Screen.MousePointer = 11
    Liste1.Expand(ListIndex) = Not Liste1.Expand(ListIndex)
    Liste1.ListIndex = ListIndex
    Liste1_Click
    Screen.MousePointer = 0
End Sub

'=========================================================
' Objekt : Liste2
' Event  : Collapse
'
' Fordert das Einklappen der Untereintraege eines Listeneintrags
'=========================================================
Private Sub Liste2_Collapse(ListIndex As Integer)

    Dim I As Long
    Screen.MousePointer = 11
    While (Liste2.Indent(ListIndex + 1) > Liste2.Indent(ListIndex))
        Liste2.RemoveItem (ListIndex + 1)
        If (ListIndex + 1) = Liste2.ListCount Then
            Liste2.PictureType(ListIndex) = 0
            Screen.MousePointer = 0
            Exit Sub
        End If
    Wend
    Liste2.PictureType(ListIndex) = 0
    Screen.MousePointer = 0
End Sub

'=========================================================
' Objekt : Liste2
' Event  : DblClick
```

```
'
' Registriert den Doppelclick auf einen Listeneintrag
'=======================================================
Private Sub Liste2_DblClick()
    Screen.MousePointer = 11
    If Liste2.Indent(Liste2.ListIndex) = 2 Then
        Call cmdhinzufuegen_Click
    Else
        Liste2.Expand(Liste2.ListIndex) = Not Liste2.Expand(Liste2.ListIndex)
    End If
    Screen.MousePointer = 0
End Sub

'=======================================================
' Objekt : Liste2
' Event  : Expand
'
' Fordert das Anzeigen aller Unterelemente eines Listenelements
'=======================================================
Private Sub Liste2_Expand(ListIndex As Integer)

    Dim theAPLSet As Object
    Dim theAVOSet As Object
    Dim theBeadatSet As Object
    Dim stmtxt As String
    Dim theLine As String
    Dim herst_id, apl_index, arbeitspla, theTyp, triplatz As String
    Dim Index As Long
    Dim I As Long

    Screen.MousePointer = 11
    Index = ListIndex

    Select Case Liste2.Indent(ListIndex)
        Case 1
            theTyp = Liste2.List(ListIndex)
            Select Case theTyp
                Case "NC"
                    If Liste2.HasSubItems(ListIndex) = False Then
                        'Die dinger muessen noch eingelesen werden
                        theLine = Liste1.FullPath(Liste1.ListIndex)
                        herst_id =LTrim(RTrim(Mid$(theLine, 1,InStr(theLine, "\") - 1)))
                        theLine = Right$(theLine, Len(theLine) - InStr(theLine, "\"))
                        arbeitspla = LTrim(RTrim(Mid$(theLine, 1,
                                        InStr(theLine, "\") - 1)))
                        theLine = Right$(theLine, Len(theLine) - InStr(theLine, "\"))
                        apl_index = theLine

                        stmtxt = "select distinct ncnr_ncnr, ver, erstelldz "
                        stmtxt = stmtxt & " from zncnrver, zstatfreigs, zstationen,
                                        zbearbei_1 where"
                        stmtxt = stmtxt & " aktver = 1 and"
                        stmtxt = stmtxt & " zstatfreigs.ncnrver_ver = zncnrver.ver and"
                        stmtxt = stmtxt & " zstatfreigs.ncnrver_ncnr_ncnr=
                                        zncnrver.ncnr_ncnr and"
                        stmtxt = stmtxt & " zstationen.station = zstatfreigs.
                                        station_station and "
                        stmtxt = stmtxt & " zbearbei_1.station_station =
                                        zstationen.station and"
                        stmtxt = stmtxt & " zbearbei_1.beapl_platznr in
                                        (select beapl_platznr from zavo_bea_1 where"
                        stmtxt = stmtxt & " avo_apl_herst_id = '" & herst_id & "' and"
                        stmtxt = stmtxt & " avo_apl_arbeitspla='" & arbeitspla & "' and"
                        stmtxt = stmtxt & " avo_apl_index = '" & apl_index & "' and"
                        stmtxt = stmtxt & " avo_apl_bak = 0 )"
                        stmtxt = stmtxt & " order by zncnrver.ncnr_ncnr"

                        Set theBeadatSet = gDatabase.DbCreateDynaset(stmtxt,0&)
                        theBeadatSet.DbMoveFirst

                        I = 1
                        While Not (theBeadatSet.EOF)
                            Liste2.AddItem Format$(theBeadatSet.Fields("ncnr_ncnr").
                                        Value, "!@@@@@@@@@@@@@@@@") &
                                        ".NC " & " " & Format$(theBeadatSet.
                                        Fields("ver").Value, "!@@@@@@@") & "
                                        " & Format$(theBeadatSet.
```

```
                                        Fields("erstelldz").Value, "dd-mmm-
                                        yy hh:mm:ss"), ListIndex+I
                        Liste2.Indent(ListIndex + I) = Liste2.Indent(ListIndex)+ 1
                        Liste2.PictureType(ListIndex + I) = 2
                        theBeadatSet.DbMoveNext
                        I = I + 1
                    Wend

                    If Liste2.HasSubItems(ListIndex) = False Then
                        Screen.MousePointer = 0
                        Exit Sub
                    Else
                        Liste2.Expand(ListIndex) = True
                    End If
                Else
                    Liste2.PictureType(ListIndex) = 1
                End If

            Case Else

                If (Not Liste2.HasSubItems(ListIndex)) Then
                    stmtxt = "select distinct bearb_nr, ver, erstelldz "
                    stmtxt = stmtxt & "from zbeaver"
                    stmtxt = stmtxt & " where aktver = 1 and"
                    stmtxt = stmtxt & " bearb_typ_typ = '" & theTyp & "'"
                    stmtxt = stmtxt & " order by bearb_nr"

                    Set theBeadatSet = gDatabase.DbCreateDynaset(stmtxt,0&)
                    theBeadatSet.DbMoveFirst

                    I = 1
                    While Not (theBeadatSet.EOF)
                        Liste2.AddItem Format$( theBeadatSet.Fields("bearb_nr").
                                        Value, "!@@@@@@@@@@@@@@@@") &
                                        "." & theTyp & " " & Format$(
                                        theBeadatSet.Fields("ver").Value,
                                        "!@@@@@@@") & " " & Format$(
                                        theBeadatSet.Fields("erstelldz").
                                        Value, "dd-mmm-yy hh:mm:ss"),
                                        ListIndex+I
                        'Liste2.AddItem theBeadatSet.Fields("nr").Value & "."
                                        & Chr(9) & Chr(9) & theTyp & Chr(9)
                                        & Chr(9) & theBeadatSet.
                                        Fields("aktver").Value &
                                        Chr(9) & Chr(9) & theBeadatSet.
                                        Fields("erstelldz").Value &
                                        Chr(9), ListIndex + i
                        Liste2.Indent(ListIndex + I) = Liste2.Indent(ListIndex) + 1
                        Liste2.PictureType(ListIndex + I) = 2
                        theBeadatSet.DbMoveNext
                        I = I + 1
                    Wend

                    If Liste2.HasSubItems(ListIndex) = False Then
                        Screen.MousePointer = 0
                        Exit Sub
                    Else
                        Liste2.Expand(ListIndex) = True
                    End If
                Else
                    Liste2.PictureType(ListIndex) = 1
                End If
            End Select
    End Select
    Screen.MousePointer = 0
End Sub

'=======================================================
' Objekt : Liste2
' Event  : PictureClick
'
' Registiert das clicken auf ein Icon in der Liste
'=======================================================
Private Sub Liste2_PictureClick(ListIndex As Integer)
    Screen.MousePointer = 11
    Liste2.ListIndex = ListIndex
    Screen.MousePointer = 0
```

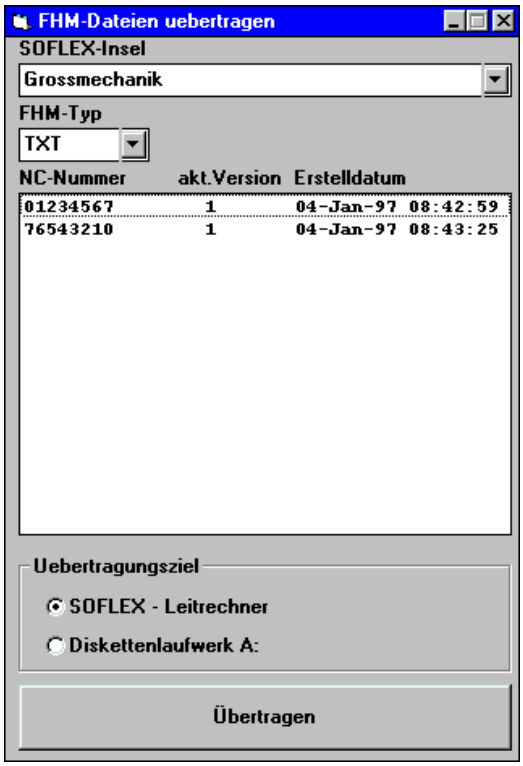
```
End Sub

'=========================================================
' Objekt : Liste2
' Event  : PictureDblClick
'
' Registriert den Doppelclick auf ein Icon in der Liste
'=========================================================
Private Sub Liste2_PictureDblClick(ListIndex As Integer)
    Screen.MousePointer = 11
    Liste2.Expand(ListIndex) = Not Liste2.Expand(ListIndex)
    Liste2.ListIndex = ListIndex
    Screen.MousePointer = 0
End Sub
```

Bildschirmabzüge verschiedener Module



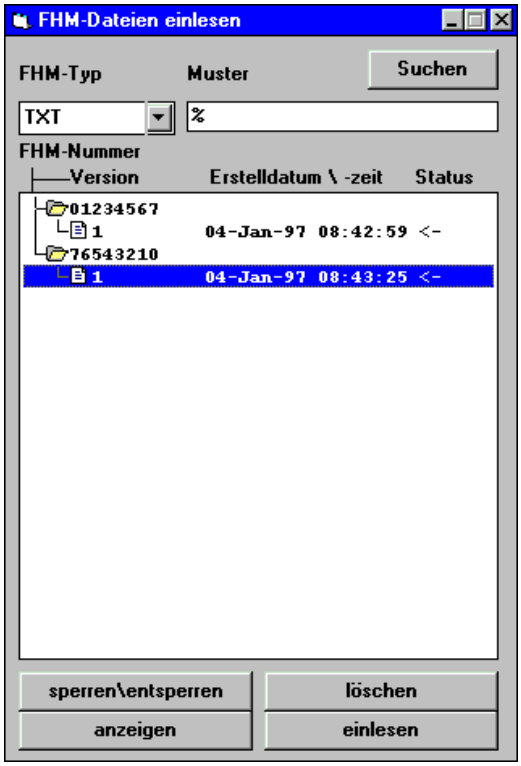
Autor des Moduls : Jörg Evermann

Anh. W.1: Bildschirmdruck Modul Fertigungshilfsmitteldaten zuordnen



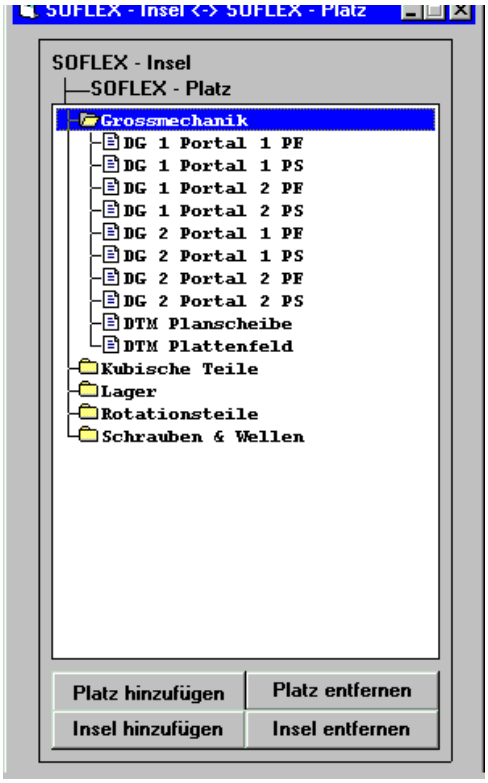
Autoren des Moduls: Jörg Evermann, ABB Kraftwerke AG

Anh. W.2: Bildschirmdruck Modul Fertigungshilfsmittel übertragen



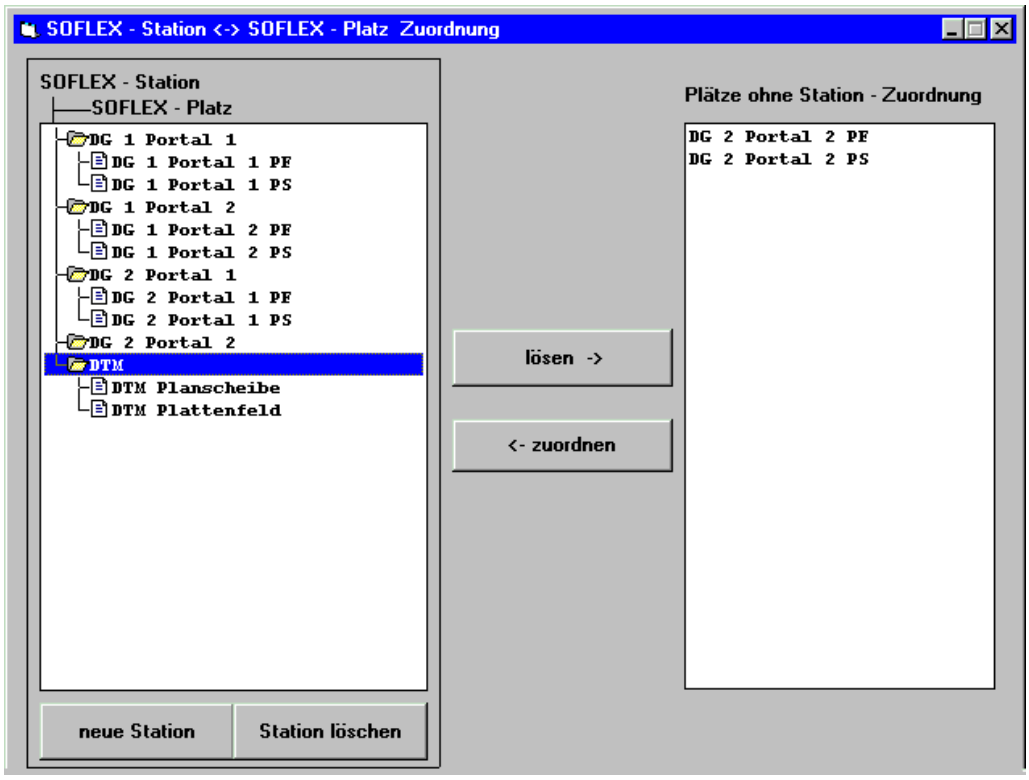
Autoren des Moduls: Jörg Evermann, ABB Kraftwerke AG

Anh. W.3: Bildschirmdruck Modul Fertigungshilfsmitteldaten lesen und verwalten



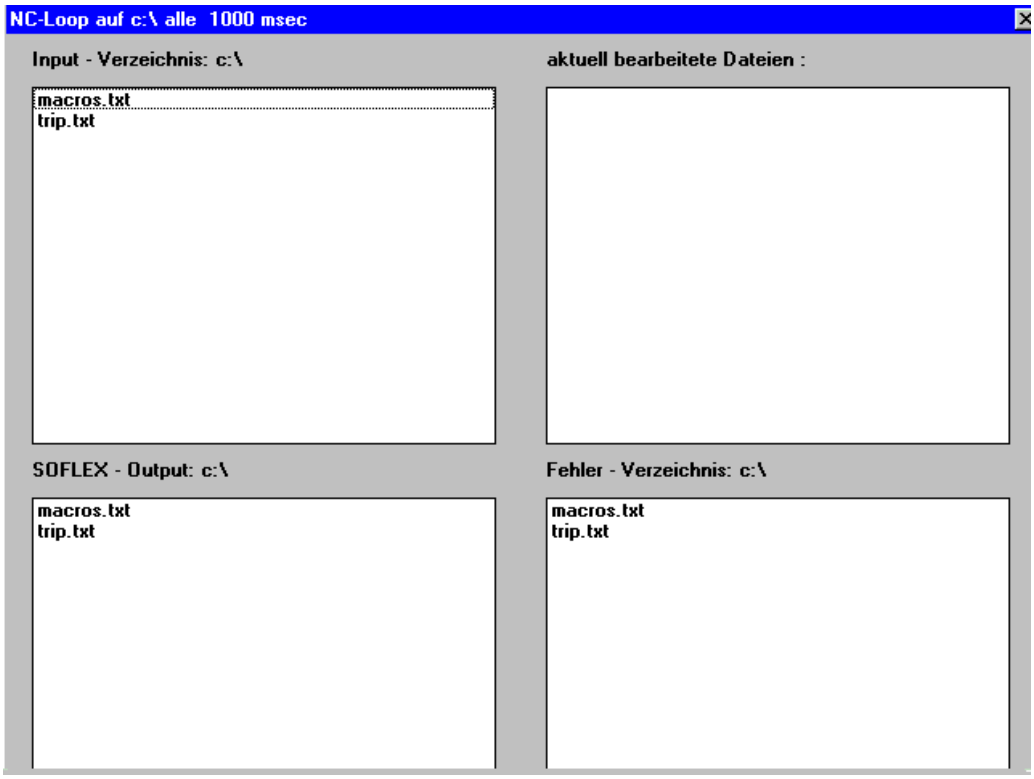
Autoren des Moduls : Jörg Evermann, ABB Kraftwerke AG

Anh. W.4: Bildschirmdruck Modul Fertigungsinsel-Bearbeitungsplatz Zuordnungen



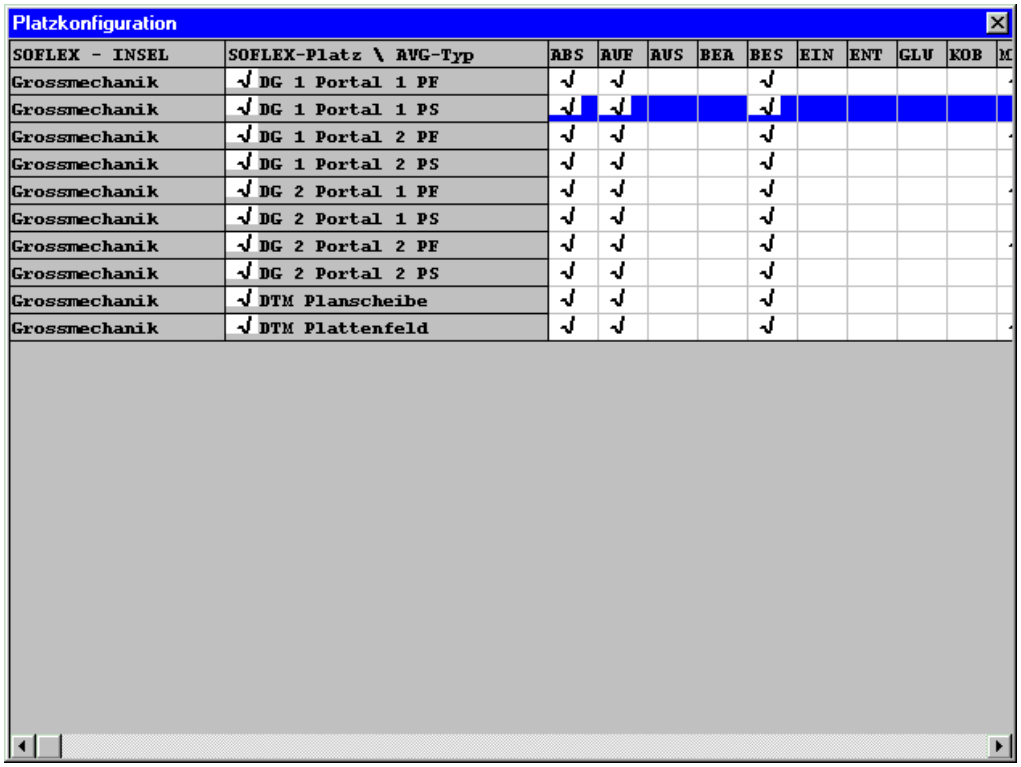
Autoren des Moduls: Jörg Evermann, ABB Kraftwerke AG

Anh. W.5: Bildschirmdruck Modul Bearbeitungsplatz-Bearbeitungsstationen Zuordnung



Autoren des Moduls: Jörg Evermann, ABB Kraftwerke AG

Anh. W.6: Bildschirmdruck Modul NC-Programme einlesen



SOFLEX - INSEL	SOFLEX-Platz \ AVG-Typ	ABS	AUF	AUS	BER	BES	EIN	ENT	GLU	KOB
Grossmechanik	DG 1 Portal 1 PF	√	√			√				
Grossmechanik	DG 1 Portal 1 PS	√	√			√				
Grossmechanik	DG 1 Portal 2 PF	√	√			√				
Grossmechanik	DG 1 Portal 2 PS	√	√			√				
Grossmechanik	DG 2 Portal 1 PF	√	√			√				
Grossmechanik	DG 2 Portal 1 PS	√	√			√				
Grossmechanik	DG 2 Portal 2 PF	√	√			√				
Grossmechanik	DG 2 Portal 2 PS	√	√			√				
Grossmechanik	DTM Planscheibe	√	√			√				
Grossmechanik	DTM Plattenfeld	√	√			√				

Autoren des Moduls: Jörg Evermann, ABB Kraftwerke AG

Anh. W.7: Bildschirmdruck Modul Bearbeitungsplatz-Bearbeitungsarten Zuordnung

Abschließende Erklärung

Ich versichere hiermit, daß ich meine Diplomhausarbeit „Prozeßorientierte Entwicklung und Implementierung eines Interface-Systems zur PPS-Leitsystemkopplung in der Inselfertigung am Beispiel der Turbinenfertigung der ABB-Kraftwerke AG, Mannheim“ selbständig und ohne fremde Hilfe angefertigt habe, und daß ich alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlegenden Ausführungen meiner Arbeit besonders gekennzeichnet und die Quellen zitiert habe.

Münster, den 27. März 1997