Burgholz, Martin; Kurbel, Karl; Nietsch, Thomas; Rautenstrauch, Claus

Working Paper

Erfahrungen bei der Entwicklung und Portierung eines elektronischen Leitstands

Arbeitsberichte des Instituts für Wirtschaftsinformatik, No. 8

Provided in Cooperation with:
University of Münster, Department of Information Systems

Suggested Citation: Burgholz, Martin; Kurbel, Karl; Nietsch, Thomas; Rautenstrauch, Claus (1992) : Erfahrungen bei der Entwicklung und Portierung eines elektronischen Leitstands, Arbeitsberichte des Instituts für Wirtschaftsinformatik, No. 8, Westfälische Wilhelms-Universität Münster, Institut für Wirtschaftsinformatik, Münster

This Version is available at:
https://hdl.handle.net/10419/59361

Arbeitsberichte des Instituts für Wirtschaftsinformatik
Herausgeber: Prof. Dr. J. Becker, Prof. Dr. H. L. Grob, Prof. Dr. K. Kurbel,
Prof. Dr. U. Müller-Funk

Arbeitsbericht Nr. 8

Erfahrungen bei der Entwicklung und Portierung eines elektronischen Leitstands

Martin Burgholz, Karl Kurbel,
Thomas Nietsch, Claus Rautenstrauch

Institut für Wirtschaftsinformatik der Westfälischen Wilhelms-Universität Münster,
Grevener Str. 91, 4400 Münster, Tel. (0251) 83-9750, Fax (0251) 83-9754
Februar 1992

Inhalt

Zusammenfassung

Der Beitrag gibt einen Bericht über Erfahrungen, die bei der Entwicklung und Portierung des elektronischen Leitstands L1 in verschiedenen Hardware- und Softwareumgebungen gewonnen wurden. Im zweiten Kapitel wird kurz die Funktionalität von Leitständen beschrieben und das zugrundeliegende Datenmodell sowie seine Repräsentation zur Laufzeit behandelt. Die chronologische Entwicklung der Konzepte und Implementierungen, angefangen mit einer ersten Version unter MS-DOS und Weiterentwicklungen unter Unix, ist Gegenstand des Hauptteils. Es werden softwaretechnische Probleme diskutiert und der Wandel des Leitstands von der einfachen Abbildung einer manuellen Plantafel bis hin zu einem vielfältigen Informationsinstrument aufgezeigt.

1 Einleitung

Die Leitstand-Entwicklungen des Instituts für Wirtschaftsinformatik begannen um 1985 an der Universität Dortmund und werden seit 1990 in Münster fortgeführt. Die erste Version des Leitstandsystems L1 entstand im Rahmen des DFG-Projekts "Interaktive Planungs- und Steuerungssysteme mit Orientierung an Arbeitsplatzrechnern und Bürokommunikationstechnologien" im Zeitraum 1986-1988. Hier wurden grundlegende neue Ideen und Anforderungen an computergestützte Instrumente für die Fertigungssteuerung entwickelt und prototypisch umgesetzt[1]. Um eine möglichst hohe Benutzerakzeptanz zu erreichen, wurden leistungsfähige Funktionen bereitgestellt, die den manuellen Vorgehensweisen bei der kurzfristigen Planung und Steuerung nachgebildet sind und den Disponenten wirkungsvoll unterstützen. Da für die Prototypentwicklung zunächst die Kapazität eines PC's

[1] Vgl. Kurbel, Meynert (1988).

unter MS-DOS ausreichend schien, wurde ein Werkzeug entwickelt, das auf grafischen Darstellungen der Fertigungssteuerung basierte. Es setzte auf einer relationalen Datenbank auf und nutzte als Fensteroberfläche MS-Windows. Die zugrundeliegenden Ideen erwiesen sich, zusammen mit den gewonnenen Erfahrungen, als tragfähige Basis für eine Weiterentwicklung in einer leistungsfähigeren Umgebung. Die Anforderungen aus der Praxis, insbesondere Mengengerüste, welche die Kapazität von MS-DOS überstiegen, und die erforderliche Einbindung in ein CIM-Umfeld legten die Implementierung unter Unix nahe. Die MS-DOS-Version wurde teils nach Unix portiert, teils reimplementiert und teils konzeptionell erweitert. Die MS-DOS- und Unix-Erfahrungen fließen gegenwärtig in ein Projekt ein, in dem auf Basis objektorientierter Softwaretechnologie und neuer Konzepte ein hochflexibles Leitstandsystem entwickelt wird.

2 L1 im Überblick

2.1 Aufgaben eines Leitstands im PPS-Umfeld

Die Defizite der gängigen Produktionsplanungs- und Steuerungssysteme (PPS-Systeme) liegen vor allem im Bereich der kurzfristigen Planung und Werkstattsteuerung. Gerade dieser Bereich gewinnt um so mehr an Bedeutung, je höher die Anforderungen an Flexibilität, hohe Termintreue und kurze Durchlaufzeiten werden. Mit dem Ziel, die Lücke zwischen der Grobplanung der PPS-Systeme auf der einen Seite und der Fertigungsfeinplanung und -steuerung auf der anderen Seite zu schließen, wurden in den letzten Jahren dedizierte Systeme entwickelt (sogenannte "grafische" oder "elektronische" Leitstände). Sie sind den manuellen Fertigungsleitständen nachempfunden und zeichnen sich durch umfangreiche Funktionalität und gute Ergonomie aus.

Aufgabe der Leitstände ist es, die Ergebnisse der Grobplanung in detaillierte Maschinenbelegungs-, Termin- und Reihenfolgepläne umzusetzen und informationstechnische Hilfestellung bei der Realisierung der Pläne zu geben. Insbesondere geht es darum, die grobterminierten Fertigungsaufträge so in die Fertigung einzulasten, daß die oben genannten Anforderungen erfüllt, Umdispositionen wirkungsvoll unterstützt und aktuelle Informationen über die Fertigungssituation bereitgestellt werden. Von entscheidender Bedeutung ist die zeitnahe Verarbeitung von Störungen (z.B. Maschinenausfall) und anderen kurzfristigen Änderungen (z.B. Eilaufträge, Stornierungen). Die heutige Beliebtheit und Verbreitung der elektronischen Leitstände in der Welt der eher schwerfälligen PPS-Systeme ist nicht zuletzt auf die erzielbare Flexibilität und Reaktionsgeschwindigkeit zurückzuführen.

2.2 Bausteine von L1

Bevor eine detaillierte Maschinenbelegungsplanung sinnvoll durchgeführt werden kann, sollte im Rahmen der *Kapazitätsdisposition* zunächst sichergestellt werden, daß die vom PPS-System übergebenen Fertigungsaufträge mit den verfügbaren Ressourcen (z.B. Betriebsmittel, Personal) überhaupt gefertigt werden können. Eine mögliche Ausgleichs-

maßnahme besteht darin, Überbelastungen und Unterbelastungen in den einzelnen Feinplanungsperioden durch geeignete Verschiebungen zu glätten ("Kapazitätsabgleich").
Die Berechnung der Kapazitätsbelastung erfolgt auf Basis der Start- und Endtermine grobgeplanter Fertigungsaufträge. Grafische Darstellungsform der Belastungsübersicht ist das *Kapazitätsgebirge*. Je nach Aggregationsgrad werden Kapazitätsangebot und Kapazitätsbedarf für Einzelbetriebsmittel, für Betriebsmittelgruppen oder für andere Kapazitätseinheiten über der Zeitachse gegenübergestellt. Die Kapazitätsbedarfe der einzelnen Arbeitsgänge werden in Form von farbigen Rechtecken in das Gebirge eingelastet. Treten Über- oder Unterbelastungen auf, können Verschiebungen interaktiv in der Grafik durchgeführt werden. Wenn dabei ein Arbeitsgang so umgeplant wird, daß die Ecktermine der Grobplanung verletzt sind, wird eine Engpaßterminierung um den umgeplanten Arbeitsgang angestoßen.
Den Kern der Funktionalität eines grafischen Leitstands bildet die *Maschinen-* oder, allgemeiner die *Betriebsmittelbelegungsplanung*. Dazu steht in L1 eine den konventionellen manuellen Plantafeln nachempfundene *elektronische Plantafel* zur Verfügung. Ihre Aufgabe ist es, den Disponenten bei der zeitlichen Zuordnung von Arbeitsgängen zu Betriebsmitteln unter Beachtung von Kapazität und Verfügbarkeit durch komfortable Funktionen zu unterstützen und die Zwischen- und Endergebnisse grafisch darzustellen.
Im Gegensatz zur Kapazitätsdisposition, bei der die Arbeitsgänge im allgemeinen noch auf einer höheren Aggregationsstufe, z.B. Betriebsmittelgruppen, behandelt werden, legt man mit der Plantafel detaillierte Termine und Auftragsreihenfolgen für die Einzelbetriebsmittel fest. Arbeitsgänge werden in Balkendiagrammen (Gantt-Diagrammen) den Ressourcen zugeordnet. Abbildung 1 gibt einen Eindruck von der verwendeten Darstellungsform.

Abb.1: Plantafel

Die wichtigsten Funktionen der Plantafel sind die automatische oder manuelle Einlastung von Arbeitsgängen, die Manipulation von Arbeitsgängen (Umplanen und Verschieben) sowie die Verarbeitung von Rückmeldungen aus der Betriebsdatenerfassung (BDE).
Im folgenden wird zunächst das Datenmodell für beide Leitstände (MS-DOS- und Unix-Version) und anschließend die Repräsentation der Daten zur Laufzeit dargestellt.

2.3 Datenmodell

Abbildung 2 zeigt das Datenmodell in Form eines Entity-Relationship-Diagramms. Die Umsetzung erfolgte in beiden Fällen mit Hilfe des relationalen Datenbanksystems Oracle. Dieses

wurde vor allem wegen der breiten Verfügbarkeit auf den verschiedensten Hardware-Plattformen sowie guter Erfahrungen mit der Stabilität gewählt. Die Schnittstelle zwischen Leitstandsoftware und Datenbanksystem wurde auf der Grundlage von "Embedded SQL" realisiert. Damit ist es möglich, den Leitstand auf alle Datenbanksysteme, die diese Schnittstelle bedienen, aufzusetzen.

Abb.2: Datenmodell des Leitstands

Die in Abbildung 2 mit Bezeichnungen versehenen Entity- und Beziehungstypen wurden in 13 Relationen abgebildet:

BM	Betriebsmittel
BMBETR	Betriebszeiten
FA	Fertigungsaufträge (Köpfe)
FABKAL	Fabrikkalender
FAG	(Fertigungs-)Arbeitsgänge
FAGBEL	Arbeitsgangbelegungen
FASTRUK	Fertigungsauftragsstrukturen
FKABEZ	Kundenauftragsbezüge
KA	Kundenaufträge
KAPOS	Kundenauftragspositionen
MATERIAL	Materialbedarfe
RUEST	Rüstzustände
SCHICHT	Schichtmodelle

Einstiegspunkt in das Datenmodell ist der Kundenauftrag (KA) bzw. die Kundenauftragsposition (KAPOS). Über die Beziehungsrelation FKABEZ wird eine n:m-Beziehung zwischen den Kundenauftragspositionen und Fertigungsaufträgen (FA) realisiert. Diese Zuordnung erlaubt es, bei der Bildung von Losen den Bezug zum Kundenauftrag zu verfolgen.
Jedem Fertigungsauftrag kann eine Menge von Materialbedarfssätzen (MATERIAL) zugeordnet werden. Darüber hinaus steht ein Fertigungsauftrag in einer n:m-Beziehung zu anderen Fertigungsaufträgen. Dieser Sachverhalt wird durch die Relation FASTRUK ausgedrückt. Die Aufgliederung eines Fertigungsauftrags in mehrere Fertigungsarbeitsgänge (FAG) wird im Datenmodell durch die 1:n-Beziehung zwischen den Entities FA und FAG deutlich. Unter den Fertigungsarbeitsgängen gibt es ebenfalls Beziehungen (z.B. Reihenfolgen, Alternativen), die hier nicht näher bezeichnet sind. Mit Hilfe der Relation RUEST lassen sich einzelnen Fertigungsarbeitsgängen verschiedene Rüstzeiten zuordnen. Die Relation FAGBEL stellt den Zusammenhang zwischen den

Fertigungsarbeitsgängen und den Betriebsmitteln (BM) dar. Einem Arbeitsgang können mehrere Betriebsmittel und einem Betriebsmittel mehrere Fertigungsarbeitsgänge zugeordnet werden.
Wann ein Betriebsmittel zur Verfügung steht, läßt sich über die Beziehungen zum Fabrikkalender (FABKAL) und zu den Schichtmodellen (SCHICHT) herausfinden. Anders als bei den Fabrikkalendern, bei denen genau einer einem Betriebsmittel zugeordnet ist, existiert eine n:m-Beziehung BMBETR zwischen den Schichtmodellen und den Betriebsmitteln.

2.4 Datenverwaltung zur Laufzeit
Zur Laufzeit werden alle Daten, die für die grafische Darstellung erforderlich sind, im Hauptspeicher gehalten. Auf diese Weise lassen sich laufzeitintensive SQL-Sequenzen und lange Antwortzeiten vermeiden. Zugriffe auf die Datenbank werden gebündelt und auf ein Minimum reduziert.
Die zur Datenverwaltung benutzten Datenstrukturen sind sortierte und unsortierte lineare Listen. Mit diesen können sowohl sachbezogene Informationen über die Objekte des Leitstands (z.B. Betriebsmittel) als auch Informationen über ihre grafische Repräsentation einfach und effizient gespeichert und manipuliert werden. Mit Hilfe generischer Datentypen lassen sich im Leitstand unterschiedlich strukturierte Daten auf einheitliche Art und Weise verwalten.
Im folgenden wird der Aufbau der beiden Listenformen beschrieben und ihre Verwendung an Beispielen gezeigt.

2.4.1 Unsortierte lineare Liste

Für Datenstrukturen, bei denen keine Ordnungsreihenfolge zu beachten ist, werden doppelt verkettete Listen verwendet. Ihre Datenelemente dürfen von beliebigem Typ sein. Die Generizität wird durch eine spezielle Verweisstruktur hergestellt, die in C-Notation folgendermaßen realisiert werden kann:

```
typedef void* LIST_DATUM;

typedef struct {
        LIST_DATUM      *Datum;
        LIST_ELEM       *Next;
        LIST_ELEM       *Prev;
} LIST_ELEM;
```

Über den Zeiger mit Namen "Datum" wird auf ein Element mit Typ "void*" verwiesen, das die Adresse von Objekten beliebigen Datentyps beinhalten kann. Der "void*"-Zeiger ("generischer Zeiger") wird dann genutzt, wenn der genaue Typ des Objekts entweder nicht bekannt ist oder während der Laufzeit variiert. Somit ist es möglich, Datenstrukturen unterschiedlicher Größe mit dem "Datum"-Zeiger zu verwalten.

Abb. 3: Generische doppelt verkettete Liste
Die Liste selbst wird durch eine separate Datenstruktur mit Zeigern auf das erste und das letzte Listenelement sowie einem Verweis auf ein Objekt beliebigen Datentyps realisiert. Das letztere kann z.B. dazu verwendet werden, Informationen über die Liste aufzunehmen. Abbildung 3 zeigt den grundsätzlichen Aufbau der Listenstruktur. In Abbildung 4 wird ihre Verwendung anhand der Verwaltung der Betriebsmitteldaten illustriert. Die "Datum"-Zeiger der Listenelemente verweisen jeweils auf eine Datenstruktur, welche die Betriebsmitteldaten und die Informationen zur grafischen Darstellung eines Betriebsmittels enthält.

Abb. 4: Betriebsmittelliste
Betriebsmittelhierarchien werden durch Verkettung der Betriebsmittel untereinander abgebildet. Das Feld "level" gibt die jeweilige Hierarchiestufe wieder. Das Gruppenflag "grpflg" wird für Betriebsmittelgruppen auf 1 gesetzt. In der Abbildung repräsentiert beispielsweise das Element ganz links eine Gruppe, der die beiden nächsten Elemente rechts angehören. Der Wert von "level" gibt die Hierarchiestufe an.

2.4.2 Sortierte lineare Liste

Die sortierte Liste wird aus der unsortierten mit Hilfe einer Vergleichsfunktion erzeugt. In der C-Notation

```
typedef struct {
        LISTE       *Liste;
        int         (*Vergleich) ( );
} SORT_LIST;
```

verweist die erste Komponente auf eine Liste, deren Elemente beim Eintragen mit Hilfe der Funktion "Vergleich" geordnet werden. Abbildung 5 verdeutlicht das Prinzip. (Der mit "LISTE" bezeichnete Kasten stellt eine Vergröberung der Ab-

bildung 3 dar). Um die Vergleichsfunktion für verschiedene Datentypen nutzen zu können, wird in der Definition von "SORT_LIST" nur auf die Adresse verwiesen. "Vergleich" liefert als Resultat einen Integerwert x:

x < 0	erstes Datum ist kleiner als zweites Datum
x = 0	beide Daten sind gleich
x > 0	erstes Datum ist größer als zweites Datum

Damit definiert sie eine totale Ordnung auf den in der Liste abgespeicherten Daten.

Abb. 5: Sortierte lineare Liste

Sortierte Listen werden beispielsweise zur Verwaltung der Fertigungsaufträge genutzt. Die einzelnen Aufträge werden nach (alphanumerischen) Fertigungsauftragsnummern sortiert. Jeder Auftrag setzt sich aus Fertigungsarbeitsgängen (FAG) zusammen, die in einer unsortierten Liste verwaltet werden. Die Arbeitsgänge besitzen jeweils einen Zeiger auf den Fertigungsauftragskopf, der alle arbeitsgangübergreifenden Daten enthält. Abbildung 6 gibt den Aufbau der Fertigungsauftragsliste (FA_LIST) wieder.

Abb. 6: Fertigungsauftragsliste

3 Realisierung unter MS-DOS

Für die erste Version des Leitstands L1 wurden als Zielgruppe kleine und mittlere Unternehmen gesehen. Daher empfahl sich die Verwendung von Standard-Technologie (PC mit Betriebssystem MS-DOS sowie Grafik-Oberfläche MS-Windows). Die MS-DOS-Version läuft auf IBM- oder IBM-kompatiblen Rechnern mit 80386-Prozessoren (oder Nachfolgern), einem Hauptspeicher von mindestens 4 MB und Festplattenkapazität von mindestens 40 MB. Für die Grafik ist ein mindestens EGA-fähiger Farbmonitor erforderlich.

3.1 Konzepte

Als Vorbild für die Gestaltung einer "elektronischen Plantafel" diente die bewährte manuelle Plantafel. Das äußere Erscheinungsbild ist sehr ähnlich. Damit sollte auch die Akzeptanz bei den Mitarbeitern gefördert werden. Die Funktionalität der elektronischen Plantafel ist jedoch deutlich höher als die einer manuellen.

Besonderes Augenmerk wurde auf komfortable Interaktionsmöglichkeiten zwischen Benutzer und System gelegt. Die grundlegende Philosophie von L1 sieht vor, zur Unterstützung des Benutzers mächtige und effiziente Planungshilfen zur Verfügung zu stellen, anstatt den Planungsvorgang zu automatisieren.

Für die Kapazitätsdisposition wird die Auslastung eines Betriebsmittels oder einer Betriebsmittelgruppe in Form von Balkendiagrammen ("Kapazitätsgebirge") dargestellt. Wie bei der Plantafel handelt es sich um eine "aktive" Grafik. Das heißt, der Benutzer kann interaktiv Manipulationen in der Grafik vornehmen, z.B. einen Kapazitätsabgleich durch Verschiebung von Arbeitsgängen durchführen. Hierbei werden Konsistenzprüfungen durchgeführt und gegebenfalls Funktionen angeboten, mit denen die Konsistenz wiederhergestellt werden kann (z.B. Neuterminierung eines Teils eines Fertigungsauftragsnetzes).

3.2 Implementierung

Die Implementierung erfolgte mit Hilfe des MS Windows Development Toolkit (Version 1.3), der Programmiersprache MS C (Version 5.0) und der Pro*C-Programmierschnittstelle von Oracle. Als grafische Benutzeroberfläche wurde MS Windows (Version 2.03) verwendet.

Kriterium für die Werkzeugauswahl waren vor allem Portabilität und Integrierbarkeit. C sowie relationale Datenbanksysteme mit SQL-Schnittstelle sind auf fast jeder Hardware-Plattform verfügbar. Die Integration des Leitstands mit anderen Softwaresystemen, die über eine SQL-Schnittstelle verfügen, ist damit unproblematisch. Bezüglich der Window-Oberfläche bestand die Hoffnung, daß bei einer Portierung lediglich ein Austausch gegen eine auf dem Zielsystem verfügbare Oberfläche (z.B. X Window unter Unix) erfolgen müsse. Diese Erwartung erwies sich jedoch als verfehlt.

3.3 Bewertung

Insgesamt war die Entwicklungsumgebung für die prototypische Implementierung durchaus tragfähig. Es mußten jedoch auch Nachteile in Kauf genommen werden. Besonders die Grenzen des Prozessors, des Betriebssystems und der PC-Implementierung des Datenbanksystems waren schnell erreicht.
Das Datenbanksystem verursacht beispielsweise unzumutbare Wartezeiten. Diese beruhen auf den zahlreichen Umsetzungsprozessen, die bei Zugriffen auf die Datenbank durchgeführt werden müssen. Das Problem wurde zum größten Teil dadurch umgangen, daß alle aktuell benutzten Daten zu Beginn einer Sitzung in den Hauptspeicher geladen und dort mit Hilfe der oben beschriebenen Listen behandelt werden. Eine erhebliche Wartezeit tritt dann nur beim Starten des Leitstands auf.
Auf diese Weise wird jedoch der ohnehin begrenzte Hauptspeicher zum Engpaß für das handhabbare Datenvolumen. Der kleine Speicherbereich von 640 KB, der von MS-DOS direkt adressiert werden kann, schränkt das Anwendungsprogramm deutlich ein. Zwar erlaubt MS-Windows die Nutzung weiterer Speicherbereiche; diese werden jedoch auch von Oracle benötigt. Für den Leitstand bedeutete dies, daß je nach Datenkonstellation insgesamt maximal ca. 400 - 600 Arbeitsgänge in der aktuellen Plantafel verwaltet werden konnten. Die Datenhaltung im Hauptspeicher beschleunigt zwar den Aufbau und Veränderungen der grafischen Darstellung erheblich. Je näher man jedoch der 640 KB-Grenze kommt, um so mehr steigen die Antwortzeiten an.
Für den Einsatz in Unternehmen mit einigen tausend Arbeitsgängen in der Planung wäre die MS-DOS-Version nicht geeignet. Wenn man auf der PC-Ebene bleiben wollte, müßte eine andere Form der Datenhaltung realisiert werden. Die neuen Windows-Versionen (3.x) mit erweiterten Speichermöglichkeiten und eine schnelle externe Speicherung in index-sequentiellen Dateien (ISAM) wären denkbare Lösungswege. Da jedoch MS-Windows nicht besonders portabel ist und für die ISAM-Speicherung keine genormte Schnittstelle (wie etwa für SQL) existiert, würde dadurch die Portabilität des Systems und die Möglichkeit, es mit anderen Systemen (z.B. PPS-System) zu integrieren, erheblich beeinträchtigt.
Probleme gab es auch hinsichtlich der Erweiterbarkeit. Während in der Anfangszeit der Leitstände bereits die reine Visualisierung der Feinplanung und der aktuellen Fertigungssituation als großer Fortschritt angesehen wurde, stiegen mit der Zeit die Benutzerwünsche. Heute werden von einem Leitstand differenzierte Planungsalgorithmen, Simulationsmöglichkeiten, die Einbindung wissensbasierter Komponenten, verschiedene Sichten und vieles mehr erwartet. Hierfür reicht ein Single-User- und Single-Tasking-Betriebssystem wie MS-DOS nicht mehr aus.
Die fehlende Multi-Tasking-Fähigkeit machte sich auch bei der Fensterverwaltung von MS-Windows bemerkbar. So ist es z.B. nicht möglich, in der Plantafel ein Fenster mit einer Erfassungsmaske für Arbeitsgang-Rückmeldungen zu öffnen, eine Rückmeldung entgegenzunehmen und die auf dem Bildschirm sichtbare Plantafelgrafik unmittelbar zu aktualisieren.

Wenn gar eine zeitnahe BDE-Anbindung oder die Koordination mehrerer dezentraler Leitstände über kommunizierende Prozesse realisiert werden soll, muß auf eine leistungsfähigere Betriebssystembasis übergegangen werden. Aus diesem Grunde wurde 1989 mit der Entwicklung einer Unix-Version des Leitstands begonnen.

4 Realisierung unter Unix

Die Unix-Basis wurde vor allem wegen der flexiblen Speicherverwaltung und der Multi-Tasking-Fähigkeiten gewählt. Außerdem sollte mit Hilfe von OSF/Motif als herstellerübergreifendem Standard die Portabilität auch bezüglich der grafischen Benutzeroberfläche verbessert werden.

Zunächst stellte sich die Frage, ob das System vollständig reimplementiert oder portiert und angepaßt werden sollte. Aufgrund der beschränkten Entwicklerkapazität wurde grundsätzlich eine Portierung angestrebt: Bei manchen Programmteilen erwies sich dies jedoch als nicht realisierbar. So verhinderten etwa die unterschiedlichen Konzepte von MS-Windows und OSF/Motif eine Portierung der Oberflächenmodule. Dagegen konnten die Datenbankzugriffe und die Verwaltung der internen Datenstrukturen sowie teilweise auch die Planungsalgorithmen mit begrenztem Aufwand auf die Unix-Plattform übertragen werden.

Die Implementierung erfolgte zunächst auf einer HP 9000/340 mit 8 MB Haupt- und 200 MB Plattenspeicher unter dem Betriebssystem HP/UX 6.5. Für die Oberflächengestaltung standen das X Window System (Version X11.3) und OSF/Motif (für HP/UX 6.5) zur Verfügung. Für die Weiterentwicklung wird derzeit eine HP/Apollo 9000/433 mit 32 MB Haupspeicher, HP/UX 7.0 und OSF/Motif 1.0 benutzt. Als Datenbanksystem kommt Oracle, Version 6.0.30, zum Einsatz.

Im folgenden werden einige Aspekte der Entwicklung unter Unix aufgezeigt. Einer der aufwendigsten Bereiche war die Realisierung der Benutzeroberfläche unter OSF/Motif. Aus diesem Grunde wird sich der nächste Abschnitt eingehender mit der Oberflächenentwicklung befassen. Er gibt einen Einblick in die eingesetzten Systeme, dient aber auch als Erfahrungsbericht für Entwickler mit ähnlichen Problemstellungen.

4.1 Grafiksystem

OSF/Motif ist eine grafische Benutzeroberfläche für die durchgängige und einheitliche Gestaltung von Benutzerschnittstellen. Sie legt die Spezifikation für das "Look and feel" einer Fensterumgebung fest. Motif baut auf dem X Window System bzw. den X Toolkit Intrinsics[2)] auf, einem Satz von Funktionen, die den Zugriff auf die Low-Level-Routinen von X Window vereinfachen und beschleunigen. Es bleibt damit unabhängig von Hardware und Betriebssystem. Motif setzt sich aus vier Teilen zusammen[3)]:

- MWM (Motif Window Manager) zum Verwalten der Fenster auf dem Bildschirm
- Widgetset zum Aufbau und zur Gestaltung von Fenstern
- UIL (User Interface Language) zur leichteren Erstellung von grafischen Oberflächen

---

[2)] Vgl. Scheiffler, Gettys (1986), S. 79 ff.
[3)] Vgl. Young (1990).

- Style Guide, Richtlinien für die Gestaltung der Oberfläche

Für den Progammierer ist der Motif-Widgetset (Widget = Window Gadget; Gadget = Ding, Apparat) besonders wichtig. Die Motif-Widgets stellen Bausteine für die Konstruktion einer Oberfläche dar. Das Widget ist eine zur Laufzeit erzeugte Struktur, die Informationen über den Zustand eines grafischen Objekts enthält. Das Widgetset von Motif umfaßt ca. 30 Widgets (z.B. Menüs, Buttons, Listboxen). Für spezielle Anforderungen, die sich mit den gegebenen Widgets nicht oder nur schwer lösen lassen, ist die Entwicklung eigener Widgets auf Basis bereits existierender Widgets möglich.
Jedes Widget gehört einer Widgetklasse an, die statisch allokiert und initialisiert wird und Operationen enthält, die für die Widgets der Klasse erlaubt sind. Applikationen werden mit Instanzen solcher Klassen geschaffen. Die Widgetklasse ist logisch die Menge aller Prozeduren und Daten, die mit allen Widgets der entsprechenden Klasse verbunden sind. Prozeduren und Daten können vererbt werden. Die aus dem objektorientierten Paradigma bekannte Technik der Vererbung macht einen hierarchischen Aufbau der Widgetklassen möglich. Eigenschaften übergeordneter Klassen können an Klassen tieferliegender Hierarchiestufen vererbt werden. Auf diese Weise brauchen Funktionen und Daten, die mehrere Widgets gemeinsam nutzen, nur an einer Stelle implementiert zu werden.

4.2 Implementierungserfahrungen

Zur Ausgabe der Widgets auf dem Bildschirm bedarf es der Routinen des X Window-Systems. Im Leitstandsystem müssen vor allem

- Betriebsmittel
- Arbeitsgänge
- Arbeitspausen
- Zeitachsenelemente (Zeitstrahl, Markierungen)

dargestellt werden. Um diese Objekte auf den Bildschirm zu bringen und sie dort sichtbar zu halten, sind zum Teil mehrere Prozeduraufrufe notwendig. Dazu zählen beispielsweise das Erzeugen des grafischen Kontexts, das Setzen der richtigen Attribute im grafischen Kontext und das eigentliche Zeichnen der Objekte. Für die erneute Darstellung nach bestimmten Ereignissen ist das Anwendungsprogramm verantwortlich. Das X Window-System sendet lediglich ein Signal an die Applikation bzw. an das entsprechende Fenster[4)].
Um ein ständiges Abarbeiten der Prozeduren beim Neuzeichnen zu vermeiden, wurde zur Darstellung einiger Bildschirmbereiche zunächst ein im Speicher abgelegtes verborgenes Fenster (Pixmap) erzeugt. In dieses werden die gesamten Objekte des Bereichs, z.B. des Zeitstrangs, gezeichnet. Die Größe des Pixmaps entspricht dem des maximal darstellbaren Informationsraums, vergleichbar etwa dem in

---

[4)] Vgl. Jones (1989) S. 29 ff.

anderen Systemen verwendeten "Weltkoordinatensystem". Die momentan relevanten Planungsinformationen werden in einem virtuellen Fenster, dem sog. XmScrolledWindow, dargestellt. Der Inhalt des Pixmaps muß dann lediglich in das XmScrolledWindow kopiert werden, ohne daß ein Aufruf von Zeichenroutinen erforderlich ist. Bei einem neuen Bildschirmaufbau - meist durch ein Verschieben des Scrollbalkens veranlaßt - genügt es, die Argumente für den Kopiervorgang mit den richtigen Werten zu versehen; der entsprechende Ausschnitt wird dann sichtbar.
Der Aufbau der Applikation unterscheidet sich deutlich von dem der MS-DOS-Version. Da im X Window-System Fenster in einer Hierarchie stehen, werden Betriebsmittel, Arbeitsgänge und Pausen, die Zeitachse und die Statuszeile jeweils in einem separaten Fenster (Widget) abgebildet. Im MS-DOS-Leitstand sind dagegen sämtliche Informationen in einem einzigen Fenster dargestellt.
Auf der Basis der Pixmap-Technik laufen der Aufbau und die Veränderung der Fenster verhältnismäßig effizient ab. Deshalb ist es möglich, im Unix-Leitstand mehrere Fenster gleichzeitig zur Verfügung zu stellen und mit aktuellen Daten zu versorgen. Der Benutzer kann beispielweise in einem Fenster die Plantafel bearbeiten und in einem anderen das Kapazitätsgebirge eines Engpaßbetriebsmittels im Auge behalten.

4.3 Bewertung

Die Programmentwicklung unter Unix und unter MS-DOS unterscheiden sich erheblich. Unix, primär als Betriebssystem für den Softwareentwickler geschaffen, bietet eine Vielzahl von Kommandos und Werkzeugen für die Programmerstellung. Dies gilt insbesondere bei Programmmierung in C, da Unix selbst in C implementiert ist. Nützlich sind die Multi-User- und Multi-Tasking-Fähigkeiten, die zur Verkürzung von Entwicklungszyklen (Codierung, Übersetzung und Fehlersuche) beitragen. Ein vergleichbares Arbeiten mit mehreren Entwicklern ist unter MS-DOS nicht möglich.
Als positiv wurde auch die Verwendung von X-Terminals empfunden. Diese stellen besonders beim industriellen Einsatz eines Leitstands eine interessante Alternative zu den weitaus teureren Workstations dar.
Die durch die Portierung erhoffte Performance-Steigerung konnte nur teilweise realisiert werden. Trotz der sehr effizienten Speicherverwaltung von Unix erwies sich die erste Entwicklungsplattform (HP 9000/340) als zu leistungsschwach. Dies lag vor allem an den hohen Speicheranforderungen des X Window-Systems. Der Übergang auf die HP/Apollo 9000/433 mit 32 MB und identischem Prozessor ergab zwar eine deutliche Leistungssteigerung, kann allerdings immer noch nicht als befriedigend bezeichnet werden. Der Grund liegt vor allem in der langsamen X-Server-Software des Betriebssystems HP UX. Hier wäre eine schnellere Version wünschenswert.
Die Portabilität bezüglich unterschiedlicher Hardware-Plattformen wird durch Unix gefördert. Grundsätzlich gilt die Portierung des Codes als unproblematisch. Dies konnte sowohl bei der Übertragung von der HP 9000/340 auf die HP

9000/433 als auch beim Prozessorwechsel auf das RISC-System HP 9000/835 festgestellt werden. Probleme bereiteten nur die unterschiedlichen Oracle-Versionen. Hier wurde eine Anpassung des Codes wegen der unterschiedlichen Präprozessoren erforderlich. Auch mußte der Bindevorgang an die erhöhte Anzahl von Bibliotheken des Datenbanksystems angepaßt werden. Schließlich erwiesen sich die Implementierungen der X-Server-Software auf den verschiedenen Hardware-Plattformen als unterschiedlich stabil.

5 Ausblick: Der objektorientierte Ansatz

Mit der Unix-Version von L1 wurde ein flexibles und leistungsfähiges Werkzeug für die Fertigungssteuerung erstellt. Vor dem praktischen Einsatz müssen jedoch die gleichen Probleme überwunden werden, denen sich jedes Leitstandsystem gegenübersieht: Da die Fertigungsorganisation von Betrieb zu Betrieb unterschiedlich ist, kann ein Leitstand "von der Stange" kaum sinnvoll eingesetzt werden. Vorher sind meist umfangreiche Anpassungsarbeiten erforderlich, damit die unternehmensindividuellen Anforderungen hinreichend berücksichtigt werden können.

Auf der Grundlage konventioneller Softwaretechnologie sind die Anpassungen sehr aufwendig und damit teuer. Die Kosten der Individualanpassung betragen oft eine Vielfaches des Preises für die Basisversion der Software. Eine Herausforderung an den Wirtschaftsinformatiker ist es deshalb, nach Wegen zu suchen, wie der überproportional hohe Anpassungsaufwand reduziert werden kann.

Das Problem wird zur Zeit am Institut für Wirtschaftsinformatik der Universität Münster im Rahmen des Projekts "Individualisierung von Leitstandsoftware auf der Basis objektorientierter Softwaretechnologie" (gefördert von der Stiftung Industrieforschung in Köln) angegangen. In dem Projekt soll auf Basis der bisherigen Erfahrungen ein konzeptionell völlig neuer Leitstand entwickelt werden. Erklärtes Ziel ist es, mit Hilfe der objektorientierten Softwarearchitektur Modifizierbarkeit und Erweiterbarkeit von vorneherein einzuplanen, so daß der enorme Anpassungsaufwand, der bei den bisherigen Leitstandsystemen anfällt, erheblich reduziert werden kann.

Das objektorientierte Paradigma mit der Definition von Klassenhierarchien, Vererbung, Verfeinerung und Polymorphismen zwingt den Entwickler geradezu, die Software hochgradig modular zu konzipieren. Individualanpassungen können dann durch Spezialisierung (= Verfeinerung bereits vorgegebener Objekte) oder durch Hinzufügen weiterer Objektklassen realisiert werden. Dies kommt vor allem der Zielgruppe mittelständischer Auftragsfertiger zugute, deren Anforderungen bezüglich maßgeschneiderter Software besonders hoch sind.

Literatur

Jones, O.: Introduction to the X Window System; Englewood Cliffs 1989.

Kurbel, K.; Meynert, J.: Flexibilität in der Fertigungssteuerung durch Einsatz eines elektronischen Leitstands; ZwF 83 (1988) 12, S. 581-585.

Scheiffler, R.W., Gettys, J.: The X Window System; ACM Transactions on Graphics 5 (1986) 2, S. 79-109.

Young, P.A.: The X Window System Programming and Applications with Xt; Englewood Cliffs 1990.

Arbeitsberichte des Instituts für Wirtschaftsinformatik

Nr. 1 Bolte, Ch.; Kurbel, K.; Moazzami, M.; Pietsch, W.: Erfahrungen bei der Entwicklung eines Informationssystems auf RDBMS- und 4GL-Basis; Februar 1991.

Nr. 2 Kurbel, K.: Das technologische Umfeld der Informationsverarbeitung - Ein subjektiver "State of the Art"-Report über Hardware, Software und Paradigmen; März 1991.

Nr. 3 Kurbel, K.: CA-Techniken und CIM; Mai 1991.

Nr. 4 Nietsch, M.; Nietsch, T.; Rautenstrauch, C.; Rinschede, M.; Siedentopf, J.: Anforderungen mittelständischer Industriebetriebe an einen elektronischen Leitstand - Ergebnisse einer Untersuchung bei zwölf Untenehmen; Juli 1991.

Nr. 5 Becker, J.; Prischmann, M.: Konnektionistische Modelle - Grundlagen und Konzepte; September 1991.

Nr. 6 Grob, H.L.: Ein produktivitätsorientierter Ansatz zur Evaluierung von Beratungserfolgen; September 1991.

Nr. 7 Becker, J.: CIM und Logistik; Oktober 1991.