Pflichtpraktikum
im Rahmen des Hauptstudiums im Studiengang Informatik
an der TU Dresden

„Simulation und Animation eines Fertigungsprozesses“

von

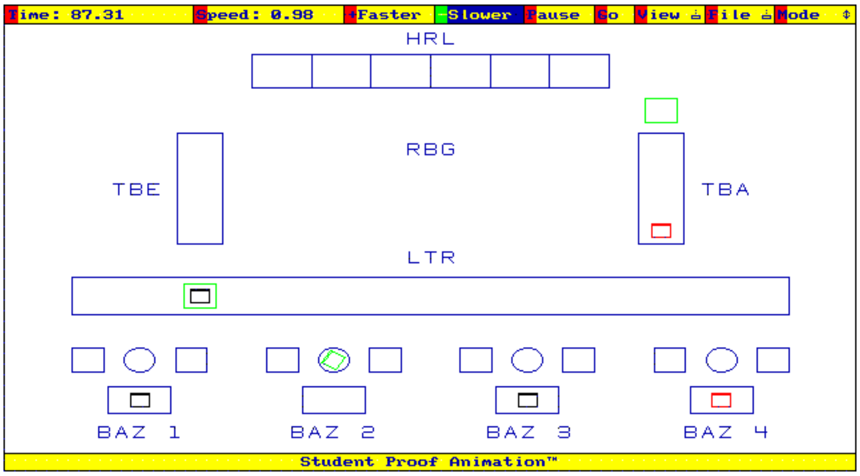
Torsten Keil
TU Dresden, Fakultät Informatik

02.02.2000

Betreuer:
Dr.-Ing. Robert Koch, TU Dresden

Dokumentation

Inhalt:



Ablaufsteuerung:

Die grobe Struktur der Ablaufsteuerung des Fertigungsprozesses kann dem folgenden Diagramm (S. 10f) entnommen werden. Genaue Details der einzelnen Belegungen von Einrichtungen bzw. Speichern zu Beschreiben würde zu umfangreich werden. Aus diesem Grund werde ich an dieser Stelle nur auf ausgewählte Stellen des Quelltextes eingehen. Um direkte Angaben zum Quelltext herstellen zu können, habe ich das durch die Simulation entstehende List-File „prakt10.lis“ verwendet, da in diesem Blöcke eindeutig gekennzeichnet sind.

Definitionsteil:

- In diesem Teil der Simulation werden Variablen deklariert, die zur Steuerung des Simulationsprozesses notwendig waren.
- ATFFILE ... Bezeichner für die Datei „prakt.atf“, die für die spätere Animation als Ablaufplan genutzt wird. Diese Datei wird während jeder Simulation neu erzeugt.
- HRL, BAZ1, BAZ2, BAZ3, BAZ4 als Bezeichner für Speicher.
- Dabei steht HRL für das Hochregallager und BAZx für die einzelnen Bearbeitungszentren, welche jeweils aus einem Drehtisch (DTx) zwei Zwischenlagern (ZWLxy) und einer Maschine (MASx) bestehen.
- Weitere Variablen nun im Überblick:

Bezeichner	Typ	Funktion
&OBJX, &OBJY	REAL	Variablen für die Übergabe von Positionen in das ATF-File
&ZEITRBG1, &ZEITRBG2, &ZEITLTR1, &ZEITLTR2	REAL	Variablen für die Steuerung von RBG und LTR
&RBGPOS, &LTRPOS	INTEGER	Variablen für die Position von RBG und LTR
&HRLBLGT	INTEGER	Variable für die Fachbelegung des HRL
&FACHX	INTEGER	Hilfsvariable für die Belegungsprozedur des HRL
MB$ZEIT	BYTE-MATRIX	Speicher-Matrix für die Zeit der einzelnen Arbeitsgänge Zeilenindex: Arbeitsgang 1-5 Spaltenindex: (BAZ1-BAZ4) * Offset Offset: Auftragsart - 1
MB$FACHAG	BYTE-MATRIX	Speicher-Matrix, ob an dem Teil in Fach x der Arbeitsgang y noch durchgeführt werden muß Zeilenindex: Fach 1-6 Spaltenindex: Arbeitsgang 1-5

Tabelle 1

Simulationsteil:

Um eine korrekte Simulation und Umsetzung der Aufgabenstellung zu der Prozesssteuerung zu erlangen, war es notwendig auf folgende Probleme zu achten:

- Es müssen sämtliche Parameter der Aufgabenstellung in das System übernommen werden. ➔ Dies wurde durch geeignete Wahl der Variablen (siehe: Tabelle 1) erreicht.
- Es dürfen innerhalb des Systems keine Verklemmungen auftreten. ➔ Dafür war es notwendig, an bestimmten Stellen des Systems mehrere Einrichtungen auf Verfügbarkeit zu testen (siehe: Probleme bei der Programmierung). An folgenden Stellen neigt das System zu Verklemmungen:
  - ♦ Teile in folgenden Einrichtungen: RBG > TBA > LTR > TBE > RBG
  - ♦ bzw. Teil1: LTR > DT > MAS und MAS > DT > ZWL
- Die geforderte Optimierung der Reihenfolge der Arbeitsgänge spielte eine wesentliche Rolle bei der Komplexität der Simulation.
- Um von der Simulation anschließend eine korrekte Animation zu erhalten, stellte sich eine korrekte Semantik von Simulationsoperationen und Animationsoperationen als notwendig heraus.

Animationsteil - Layoutfile:

Bei der Erstellung des Layoutfiles konnte durch geschickte Wahl der Lage innerhalb des Koordinatensystems eine nicht unwesentliche Erleichterung bei der Animation erreicht werden:

- Der LTR liegt direkt auf der x-Achse.
- Das System wurde symmetrisch zur y-Achse angelegt.
- Die Elementklassen wurden zentralsymmetrisch deklariert.

Probleme bei der Programmierung:

- Um zu erreichen das keine Verklemmungen während der Ablaufsteuerung auftreten, mussten mehrere Einrichtungen auf Verfügbarkeit getestet werden. Bsp.: Sollte ein Teil mit dem RBG zum TBA transportiert werden, so ist es notwendig, vorher abzutesten ob neben dem RBG auch das TBA verfügbar ist. (Sonst kann oben beschriebener Fall der Verklemmung auftreten.) Dies sollte sich einfach durch folgende Anweisung realisieren lassen:

```
GATE    FNU     RBG
GATE    FNU     TBA
SEIZE           RBG
```

  Aus der Kombination dieser beiden Anweisungen folgt richtig, das das Teil erst weiterrückt, wenn RBG und danach TBA frei sind. Bevor also TBA verfügbar ist, kann RBG schon belegt sein.
  Die zweite Überlegung war beide durch einen Zyklus zu „verbinden“:

```
MARKE   GATE    FNU     RBG
        GATE    FNU     TBA,MARKE
        SEIZE           RBG
```

  Die ernüchternde Konsequenz war, das sich das System in einer Endlosschleife verfing, sobald RBG frei, TBA aber belegt war, da ja zwischen den Durchläufen der Schleife keine Zeit „verging“.
  Also mussten aufeinanderfolgende Durchläufe verzögert werden:

```
MARKE1      GATE        FNU     RBG
            GATE        FU      TBA,MARKE2
            ADVANCE             0.1
            TRANSFER            ,MARKE1
MARKE2      SEIZE               RBG
```

  Diese Lösung des Problems findet auch in der Simulation Anwendung (Blöcke 26ff). Man kann dies auch eleganter (?) durch relative Sprünge lösen, jedoch habe ich in Hinsicht auf Übersichtlichkeit darauf verzichtet.
- Ein zweiter Punkt der einige Probleme bereitete, bestand in der Umsetzung der organisatorischen Reihenfolge (den Regeln nach denen die BAZ’s zu belegen waren). Die Forderung bestand darin, das vorrangig freie Maschinen zu belegen waren. Anschließend wurden möglichst kurze Transportwege gefordert.
  Dies flexibel zu gestalten, war mit den vorhandenen GPSS/H Befehlen nicht zu realisieren. So konnte man zwar durch Sprünge zu den Verschiedenen Elementen/Marken der Steuerung verzweigen. Jedoch nur innerhalb eines starren, durch den Programmierer vorgegebenen, Schemas.

Folgende Vorgehensweise löste die Probleme:
(1) Jedes Teil weiß, wo am LTR es sich befindet. (Blöcke 64ff)
(2) Darauf hin lässt sich eine Routine zum verzweigen entwickeln, die freie Maschinen ihrer räumlichen Nähe nach ansteuert. (Blöcke 69-73 u. 74-133)
(3) Sollte keine freie Maschine vorhanden sein, wird ein freies BAZ nach räumlicher Nähe ausgewählt. (Blöcke 134-138 u. 139-198)
Dafür ist es aber notwendig sich eine Rücksprungadresse zu „merken". In GPSS/H gibt es jedoch keine Funktion um zu erfahren, in welchem Block sich ein Teil gerade befindet. Da aber in der vorliegenden Situation ein linearer Ablauf bei der Auswahl der Maschine stattfindet, kann man sich unter zu Hilfenahme der relativen Sprungadressen eine Rücksprungadresse generieren und diese in einem Parameter sichern.
Bsp.: Bei einer relativen Sprungadresse TRANSFER ,*+2 wird zwei Blöcke nach den aktuellen Block gesprungen (TRANSFER ,*-2 dementsprechend zwei Blöcke vor den aktuellen Block). Zum Speichern einer Adresse, die zwei Blöcke weiter vorn liegt, kann man folgenden Aufruf verwenden:

```
ASSIGN          BACK,*+2,,PH
GATE     FU     MAS3,MARKE1
ASSIGN          BACK,*+2,,PH
GATE     FU     MAS4,MARKE2  usw. (Blöcke 74ff).
```

- Ein weiteres Problem, welches aus der flexiblen Steuerung resultiert aber auch in der Struktur des Systems liegt, tritt bei folgender Situation zu Tage:
  BAZ3 und BAZ4 sind voll. Teil1 befindet sich in BAZ3 und hat nur noch einen Arbeitsgang der nur in BAZ4 erledigt werden kann. Teil2 befindet sich in BAZ4 und hat nur noch einen Arbeitsgang der nur in BAZ3 erledigt werden kann. Teil1 muss BAZ3 verlassen und belegt dadurch DT3 und LTR kann aber BAZ4 nicht betreten.
  In diesem Fall tritt wieder eine Verklemmung auf. Diese kann vielleicht durch komplett andere Ablaufsteuerung vermieden werden, tritt aber in ähnlicher Form wieder auf, wenn beide Teile in BAZ3 in BAZ4 müssen und beide Teile in BAZ4 in BAZ3 müssen. Hier wäre der einzige Ausweg ein zusätzliches Zwischenlager parallel zu den BAZ's oder eine Zwischenlagerung im HRL, welche die Effektivität drastisch senken würde.