Procescalculus bij modelleren van job-produktie fabrieken

Procescalculus bij
job-produktie f

De auteurs gebruiken de procescalculus om fabrieken met job-produktie te modelleren. In het eerste model wordt een eenvoudige fabriek gebouwd, bestaande uit één bewerkingscentrum met drie bewerkers. Het tweede model bevat dezelfde drie bewerkers, maar nu voorzien van buffers om blokkering van het systeem te voorkomen. In het derde model wordt de relatie bepaald tussen de materiaal-doorlooptijd en de materiaal-throughput als functie van de hoeveelheid onderhanden werk. Deze relatie wordt weergegeven in de zogenaamde bedrijfskarakteristiek. In het vierde model worden informatiestromen in het model geïntroduceerd. Tenslotte wordt een algemeen model gepresenteerd van een job-produktiesysteem.

In een vorig artikel [Rooda, Arentsen, 1991] zijn verschillende modellen gepresenteerd van fabrieken met een flow-produktie. In dit artikel worden verschillende modellen gepresenteerd van fabrieken met een job-produktie. Bij flow-produktie wordt een produkt tijdens zijn bewerkingen slechts eenmaal op een machine bewerkt. Bij job-produktie daarentegen kan een produkt meerdere keren op dezelfde machine worden bewerkt. Een machinefabriek en een fabriek voor de vervaardiging van integrated circuits (chips) zijn voorbeelden van fabrieken met job-produktie.
In dit artikel zullen met behulp van de procescalculus verschillende modellen van systemen met een job-produktie worden gemodelleerd om de zogenaamde bedrijfskarakteristiek van dergelijke systemen te bepalen. Hiermee is het mogelijk om vooraf inzicht in de vereiste produktiemiddelen en de gewenste besturing te verkrijgen. Met name bij complexe job-produktie systemen, zoals bijvoorbeeld een zogenaamd „flexible manufacturing system (FMS)" biedt het gebruik van procescalculus voordelen.

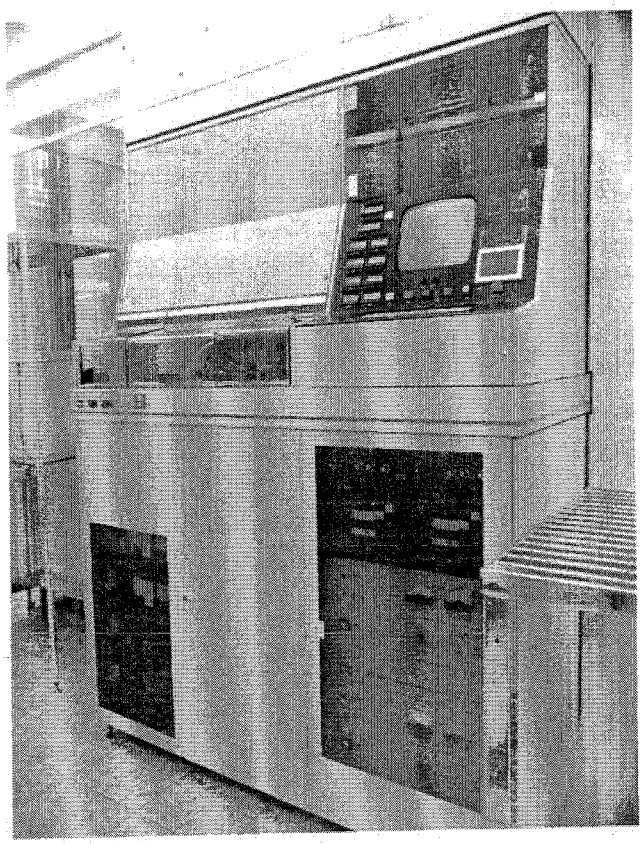
Een bewerkingscentrum

Als eerste model van een job-produktie fabriek beschouwen we een systeem waarin produkten worden vervaardigd door een bewerkingscentrum. Het bewerkingscentrum bevat drie bewerkers en een transporteur. Door

Prof.dr.ir. J.E. Rooda,
Dr.ir. J.H.A. Arentsen,
Ir. G.H. Smit,
TU Eindhoven

ij modelleren van e fabrieken

iedere bewerker wordt een andere bewerking uitgevoerd, te weten a, b en c.
Er wordt verondersteld dat ieder produkt drie (niet altijd verschillende) bewerkingen ondergaat. In totaal zijn er 27 verschillende bewerkingsvolgordes mogelijk (aaa, aab, aac, aba, abb, abc, ..., cba, cbb, cbc, cca, ccb, ccc). Er wordt aangenomen dat de bewerkingen even vaak plaatsvinden.
De tijd van iedere bewerkingsstap is gemiddeld even lang, is uniform verdeeld en ligt voor iedere bewerker tussen de 1 en 3 minuten. Er wordt in dit model geen rekening gehouden met omsteltijden, wisselen van gereedschappen, transporttijden en storingen.
Iedere drie minuten wordt door een toeleverancier nieuw materiaal bij de fabriek aangeboden. Deze toevoer kan in beginsel door het systeem worden verwerkt, daar het systeem gemiddeld per 2 minuten 1 produkt kan afleveren.



Met behulp van de procescalculus wordt een model van deze fabriek opgesteld om vast te stellen of het aanbod inderdaad door dit systeem kan worden verwerkt. Figuur 1 toont het model van de fabriek en zijn omgeving. De leverancier, de fabriek en de afnemer zijn weergegeven door de (blad-)processor Leverancier, de (geëxpandeerde) processor Fabriek en de (blad-)processor Afnemer. Voor de definities van de gebruikte begrippen wordt verwezen naar [Rooda, 1991c].

De formele beschrijving van de Afnemer luidt:

```
Processor Afnemer
body
    | produkt |
  produkt ← self receiveFrom:
  „in"
```

De Afnemer is dus altijd in staat om de geassembleerde produkten te ontvangen.

De Leverancier verstuurt de materialen. Er wordt aangenomen dat de Leverancier aangeeft wat de bewerkingsvolgorde van ieder materiaal is. Dit wordt op de volgende wijze gemodelleerd. Aan het materiaal is een bon bevestigd waarop de verschillende bewerkingsvolgordes staan vermeld: de informatie wordt (impli-

Literatuur

Kettner H., Bechte W., 1981, Neue Wege der Fertigungssteuerung durch belastungsorientierte Auftragseingabe, VDI - Z 123 (11), 459 466.

Little J.D.C., 1961, A proof for the queueing formula L = λ W, Oprns. Res. 9, 383 - 387.

Rooda J.E., 1991a, Procescalculus 1. Indeling van industriële systemen, I2 Werktuigbouwkunde 7(5), 13 - 15.

Rooda J.E., 1991b, Procescalculus 2. Systemen, modellen en geschiedenis van de procescalculus, I2 Werktuigbouwkunde 7(7), 36 - 39.

Rooda J.E., 1991b, Procescalculus 3. Definities en begrippen, I2 Werktuigbouwkunde 7(10), 35 - 40.

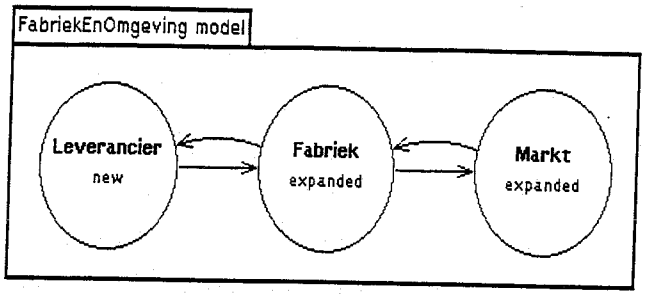
Rooda J.E., Arentsen J.H.A., 1991, Procescalculus 4. Modelleren van flow-produktie fabrieken, Mechanische Technologie 12/91, 10-20.

Smit G.H., 1992, The modelling of job-shop production facilities, Dissertatie, Technische Universiteit Eindhoven.

Wiendahl, H.-P., 1987, Belastungsorientierte Fertigungssteuerung, Carl Hanser Verlag, München, Wenen.

Prof.dr.ir.J.E. Rooda is hoogleraar Werktuigbouwkunde aan de Technische Universiteit Eindhoven.
Dr.ir.J.H.A.Arentsen is gastdocent aan de faculteit der Werktuigbouwkunde aan de Technische Universiteit Eindhoven.
Ir.G.H.Smit is technisch-wetenschappelijk medewerker aan de faculteit der Werktuigbouwkunde aan de Technische Universiteit Eindhoven.

Fig. 1. Het model van de fabriek en zijn omgeving.



ciet) met het materiaal mee verstuurd. Deze informatie kan men beschouwen als een produktgeleidebon die op het materiaal is bevestigd. Indien door een bewerker een bewerking is uitgevoerd, dan wordt deze bewerking verwijderd van de bon. Nadat de drie bewerkingen hebben plaatsgevonden is de bon dus afgewerkt. Op dat moment is het produkt gereed en kan uit het bewerkingscentrum worden gehaald. Voor het modelleren van deze bon met opdrachten wordt gebruik gemaakt van de geordende verzameling („OrderedCollection"). In [Rooda, Arentsen, 1991] is deze geordende verzameling behandeld.

De formele beschrijving van het materiaal luidt:

Object Materiaal
instance variable: bon

metOpdrachten: anOrderedCollection
↑ (self new) bon: anOrdered Collection

bon: eenOpdrachtenlijst
bon ← eenOpdrachtenlijst

bon
↑ bon

zijnAlleOpdrachtenAfgewerkt
↑ bon size = 0

verwijderOpdracht
bon removeFirst

Nieuw materiaal wordt aangemaakt met de opdracht „metOpdrachten:". Als argument wordt een bon met de te bewerken opdrachten meegegeven. De bon wordt initieel gevuld met de opdracht „bon:", terwijl de bon kan worden geïnspecteerd met de opdracht „bon".

De opdrachten:
bon ← OrderedCollection new.
bon add: „a".
bon add: „b".
bon add: „a".
blok ← Materiaal metOpdrachten: bon
leveren een nieuw blok materiaal met een bon op. In deze context dienen op dit blok de bewerkingen „a", „b"en „a" achtereenvolgens te worden uitgevoerd.
Voor de volledigheid volgt hierna de tweede mogelijkheid om een nieuw blok materiaal-met-bon te maken.

De opdrachten:
blok ← Materiaal new.
bon ← OrderedCollection new.
bon add: „a".
bon add: „b".
bon add: „a".
blok ← blok bon: bon
leveren nu eerst een nieuw blok materiaal op waaraan pas daarna een bon met de opdrachten „a", „b" en „a" wordt bevestigd.

Na de voorgaande voorbereiding, wordt de formele beschrijving van de Leverancier:

Processor Leverancier
instance variables: „bew1 bew2 bew3"

initializeTasks
bew1 ← SampleSpace data: #(„a" „b" „c").
bew2 ← SampleSpace data: #(„a" „b" „c").
bew3 ← SampleSpace data: #(„a" „b" „c")

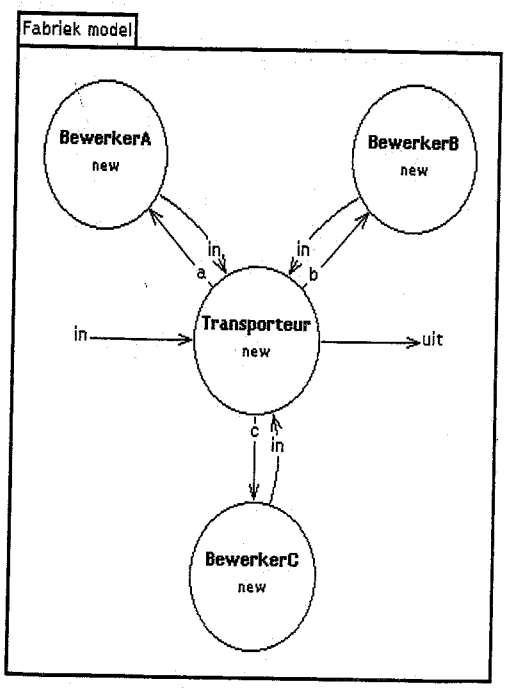
body
| bon materiaal |
bon ← OrderedCollection new.
bon add: bew1 next.
bon add: bew2 next.
bon add: bew3 next.
materiaal← _ Materiaal metOpdrachten: bon.
self send: materiaal to: „uit".
self workDuring: 3 minutes

De processor Leverancier bevat drie instance variabelen. Iedere variabele wordt initieel gekoppeld aan een zogenaamde random-generator. Het aanroepen van een van deze verdelingen, door bijvoorbeeld bew1 next, zal de waarde „a", „b"of „c" tot gevolg hebben. Deze drie trekkingen worden samengevoegd in een geordende verzameling. Deze verzameling wordt als bon aan het materiaal meegegeven. Zo'n bon kan bijvoorbeeld luiden „a" „b" „a". Het materiaal met bijbehorende bon wordt vervolgens via de „uit"-poort verzonden naar de Fabriek.

De geoefende Procescalculus-gebruiker kan de body van de Leverancier, met hetzelfde effect, verkort beschrijven door:

body
self send:
(Materiaal metOpdrachten:
(OrderedCollection
with: bew1 next
with: bew2 next
with: bew3 next))

Fig. 2. Het model van de fabriek



to: „uit".
self workDuring: 3 minutes

Na de modellering van de Afnemer en de Leverancier volgt de Fabriek (figuur 2). De Fabriek bevat een transporteur en drie bewerkers. Deze zijn gemodelleerd door de vier (blad-)processoren Transporteur, BewerkerA, BewerkerB en BewerkerC. Deze Bewerkers zijn soortgenoten.

De Transporteur probeert materiaal te ontvangen. Dit materiaal kan afkomstig zijn van de Leverancier of van één van de Bewerkers. Indien het materiaal van de Leverancier afkomstig is, dan zal het materiaal naar één van de drie Bewerkers worden getransporteerd. Welke Bewerker wordt bezocht hangt af van de eerste opdracht die op de bij het materiaal behorende bon staat. Er wordt verondersteld dat het transport tijdloos plaatsvindt.

De formele beschrijving van de Transporteur luidt:

Processor Transporteur

body
| materiaal |
materiaal ← self receiveFrom: „in".
materiaal zijnAlleOpdrachtenAfgewerkt
ifTrue: [self send: materiaal to: „uit']
ifFalse: [self send: materiaal
to: materiaal bon first]

Indien de materiaal-bon geen opdrachten meer bevat (materiaal zijnAlleOpdrachtenAfgewerkt), dan zijn alle opdrachten verwerkt en kan het materiaal het systeem verlaten. Zolang er nog opdrachten zijn, dan wordt het materiaal verstuurd naar poort „a", „b" of „c": de naam van de poort komt overeen met de opdracht die aan de kop van de bon staat. Poort „a" is verbonden met BewerkerA, poort „b" is verbonden met BewerkerB en poort „c" is verbonden met BewerkerC.

Een Bewerker probeert het materiaal te ontvangen. Gedurende ongeveer 2 minuten wordt het materiaal bewerkt. Dit wordt beschreven door een „workDuring:"- opdracht, waarbij het argument een uniforme verdeling is met een minimum en maximum waarde van 1, respectievelijk 3 minuten. Vervolgens wordt deze opdracht op de materiaal-bon verwijderd (materiaal verwijderOpdracht) om aan te geven dat deze bewerking heeft plaatsgevonden. Tenslotte wordt het materiaal weer naar de Transporteur gestuurd.

De formele beschrijving van de Bewerker luidt:

Processor Bewerker
instance variable: bewerkingsTijd

initializeTasks
bewerkingsTijd ← Uniform from: 1 minutes to: 3 minutes

body
| materiaal |
materiaal ← self receiveFrom: „in".
self workDuring: bewerkingsTijd next.
materiaal verwijderOpdracht.
self send: materiaal to: „uit"

Nu de structuur van het model en alle processor-beschrijvingen gegeven zijn is het mogelijk om dit model met behulp van de procescalculus-simulator op zijn prestatie te onderzoeken. Na enige tijd blijkt dat het model „deadlock" bevat: er treedt een blokkering op van de Transporteur. De oorzaak van deze blokkering is gemakkelijk in te zien. De Transporteur probeert bijvoorbeeld een produkt naar BewerkerB te verzenden, terwijl BewerkerB op zijn beurt probeert een (deels) gereed produkt naar de Transporteur te verzenden: het systeem komt vast te zitten. Deze blokkering is op te heffen door een

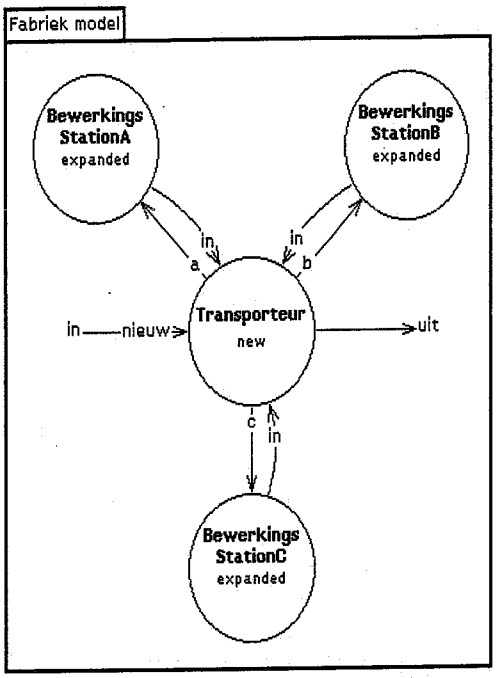


Fig. 3. Het model van de fabriek (met buffers)

Buffer voor de Transporteur te plaatsen. Een andere oplossing is om voor iedere Bewerker een Buffer te plaatsen. In het volgende model is gekozen voor de laatste oplossing.

Een bewerkingscentrum met buffers

Het aangepaste model, waarbij iedere Bewerker is vervangen door een BewerkingStation, wordt beschreven door de figuren 3 en 4. Ieder BewerkingsStation bevat een Buffer en een Bewerker. De beschrijving van de Buffer kan worden gevonden in [Rooda, Arentsen, 1991]. De beschrijving van de overige processoren blijft overeenkomstig de bovenvermelde beschrijving. Met behulp van de simulator is vast te stellen dat de blokkering is verwijderd. Het blijkt dat iedere 300 minuten ongeveer 100 produkten worden bewerkt. Door toevoeging van buffers is nu een eenvoudig maar werkend model

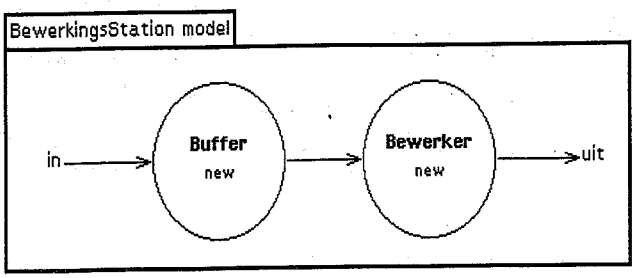


Fig. 4. Het model van een bewerkingsstation

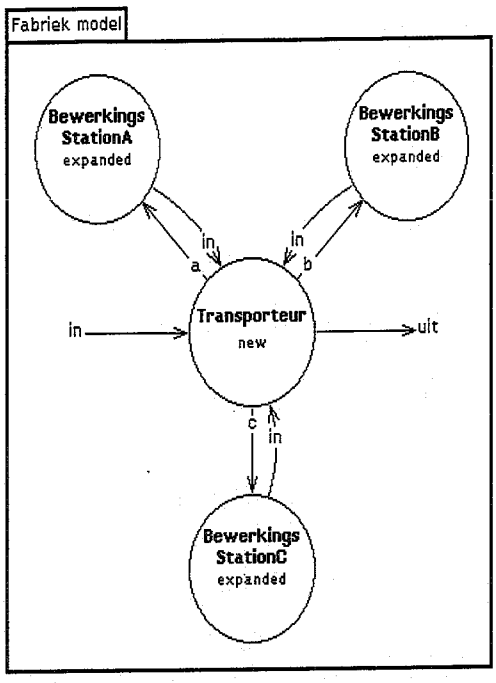


Fig. 5. Het aangepaste model van de fabriek (met buffers)

ontstaan. (Tevens zorgen de buffers voor afvlakking van het uniform verdeelde gedrag van de Bewerkers.) Dit model is dus in staat om de aangeboden materiaalstroom te verwerken.
Doorgaans is men onder andere geïnteresseerd in een zo hoog mogelijke opbrengst van de BewerkingsStations. Deze opbrengst wordt vanaf nu „materiaal throughput" genoemd en uitgedrukt in produkten per tijdseenheid. Men wil dus graag een zo hoog mogelijke materiaal throughput. Het dilemma hierbij is echter dat een hoge materiaal throughput van de BewerkingsStations tot gevolg heeft dat de wachttijden van de deels bewerkte produkten toenemen, waardoor de doorlooptijd van de produkten oploopt. Met andere woorden: wij zijn geïnteresseerd in een relatie tussen materiaal throughput en materiaaldoorlooptijd.

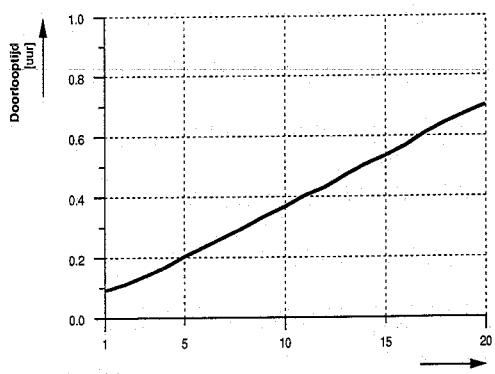


Fig. 6. De relatie tussen wip en doorlooptijd

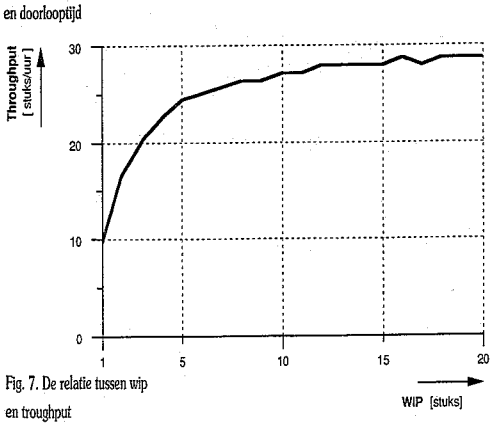


Fig. 7. De relatie tussen wip en troughput

Bedrijfskarakteristiek van een bewerkingscentrum

Om de relatie te beschrijven tussen de throughput en de doorlooptijd die in een dergelijk systeem een rol spelen, introduceren wij een zogenaamde bedrijfskarakteristiek. In een bedrijfskarakteristiek worden de throughput en de doorlooptijd bepaald als functie van de hoeveelheid onderhanden werk. Het blijkt dat de hoeveelheid onderhanden werk (Eng. work-in-process, afgekort: „wip") zeer geschikt is als parameter om het systeem te sturen [Wiendahl, 1987]. In dit artikel is de hoeveelheid onderhanden werk gedefinieerd als de totale hoeveelheid produkten die zich in het systeem bevindt. Voor een uitgebreide behandeling van het begrip onderhanden werk wordt verwezen naar de hierboven aangehaalde literatuur.
Voordat nu wordt overgegaan tot het bepalen van deze bedrijfskarakteristiek van het eerder beschreven bewerkingscentrum wordt het model aangepast ten behoeve van de zogenaamde wip-sturing.

De eerste aanpassing is dat nieuwe materialen in het systeem worden gebracht door de poort „nieuw" van de Transporteur, figuur 5. Vervolgens worden de beschrijvingen van de Leverancier en de Transporteur aangepast. In het nieuwe model vraagt de Fabriek namelijk zelf om produkten. De strategie hierbij is dat het aantal produkten in het systeem constant moet blijven: de hoeveelheid onderhanden werk blijft constant. Na een zekere initiële aanloop zal het systeem, middels de Transporteur, iedere keer om nieuw materiaal vragen nadat een gereed produkt het systeem heeft verlaten: het systeem trekt de materialen als het ware uit de Leverancier: een „pull"-systeem.

De beschrijving van de Leverancier wordt nu aangepast, zodat alleen materiaal wordt verstuurd als daaraan behoefte is:

```
Processor Leverancier
body
    self send:
        (Materiaal metBon:
            (OrderedCollection
                with: bew1 next
                with: bew2 next
                with: bew3 next))
    to: „uit"
```

Nu volgt de aanpassing van de Transporteur. Initieel moet de Transporteur het systeem beladen met een vooraf ingestelde hoeveelheid onderhanden werk. Pas nadat een gereed produkt is afgeleverd, mag de Transporteur om een nieuw blok materiaal vragen.

De formele beschrijving van de Transporteur luidt nu:

```
Processor Transporteur

initialActions
    | materiaal wip |
    wip ← ?.
    wip timesRepeat:
        [materiaal ← self receive-
        From: „nieuw".
        self send: materiaal to: ma-
        teriaal bon first]

body
    | materiaal |
    materiaal ← self receiveFrom:
    „in".
    materiaal size = 0
        ifTrue:
            [self send: materiaal to:
            „uit".
            materiaal ← self receive-
            From: „nieuw"].
    self send: materiaal to: mate-
    riaal bon first
```

Initieel zorgt de Transporteur er

met „initialActions" voor dat materiaal in de verschillende BewerkingsStations wordt gebracht. Het aantal stuks materiaal wordt aangegeven door de parameter wip. Indien wip ← 1 dan bevat het bewerkingscentrum 1 blok materiaal, wip ← 10 houdt in dat 10 blokken materiaal initieel naar de (Buffers van de) verschillende BewerkingsStations worden getransporteerd. Wanneer een blok materiaal alle bewerkingsstappen heeft doorlopen, dan wordt dit blok naar de Afnemer verstuurt, terwijl een nieuw blok via de poort „nieuw" van de Leverancier wordt betrokken.

Het is te verwachten dat het systeem minimaal gemiddeld 6 minuten (3 x 2 minuten) nodig heeft om een produkt volledig te bewerken. Het blijkt met behulp van de simulator dat, wanneer wip ← 1, het ongeveer 6 minuten duurt voordat een blok is behandeld. Tevens blijkt dat in circa 6000 minuten 1000 blokken door het systeem zijn bewerkt. Dit sluit goed aan bij de verwachting.
Vervolgens blijkt dat, wanneer wip ← 2, het ongeveer 7.4 minuten duurt voordat een blok is behandeld. Hierbij zijn circa 3700 minuten nodig om 1000 blokken te bewerken.
En bij wip ← 3 duurt het gemiddeld 9.0 minuten om een blok te bewerken, terwijl circa 3000 minuten nodig zijn voor 1000 blokken.
Bij wip ← ∞ (wip is oneindig) zullen 2000 minuten nodig zijn om 1000 blokken te bewerken.
De tijd dat een blok in het systeem doorbrengt, de doorlooptijd, kan nu als een functie van de wip worden uitgezet. Dit wordt getoond in figuur 6. De throughput van het systeem, het aantal bewerkte blokken per uur, kan ook als functie van de wip worden uitgezet. Dit wordt getoond in figuur 7. Dergelijke figuren vormen de bedrijfskarakteristiek van een systeem.

De keuze van een bepaalde wip is een strategische keuze: hoe groter de wip, des te meer objecten per uur zullen worden bewerkt, maar des te langer zal de doorlooptijd zijn. Door vergroting van het aantal produktiemiddelen kan uiteraard de ligging van deze bedrijfskarakteristieken worden verbeterd, maar tegelijkertijd worden de investeringen in produktiemiddelen hoger.

Dit model illustreert het principe om de wip als parameter te gebruiken voor het regelen van de doorlooptijd en de throughput. Om in de praktijk een soortgelijke regeling te realiseren, is bovenbeschreven model echter te eenvoudig. In de praktijk bepaalt de markt namelijk bijna altijd de behoefte aan nieuwe produkten. Daarnaast is de arbeidsinhoud van ieder produkt meestal verschillend: verschillende produkten hebben verschillende bewerkingstijden. Ook de bewerkingen zijn meestal verschillend: produkten maken gebruik van verschillende produktiemiddelen. Tevens spelen omsteltijden en transporttijden een rol. Ook het beperkt beschikbaar zijn van gereedschappen en matrijzen kan een rol spelen. In zulke gevallen is het

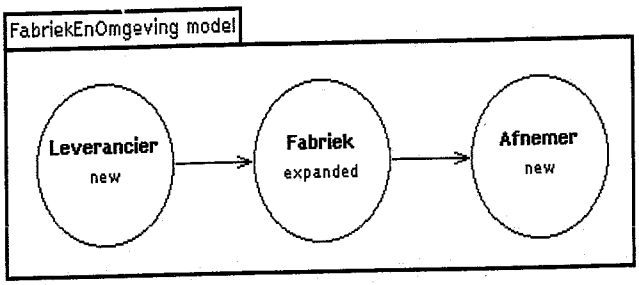


Fig. 8. Het model van de fabriek en zijn omgeving

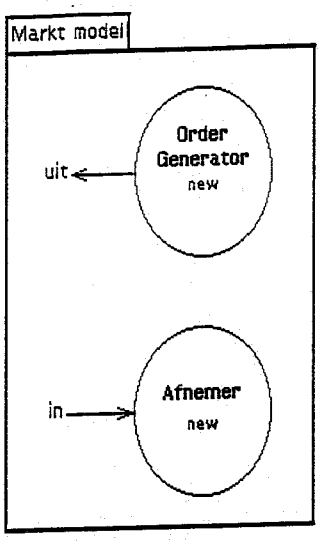


Fig. 9. Het model van de markt

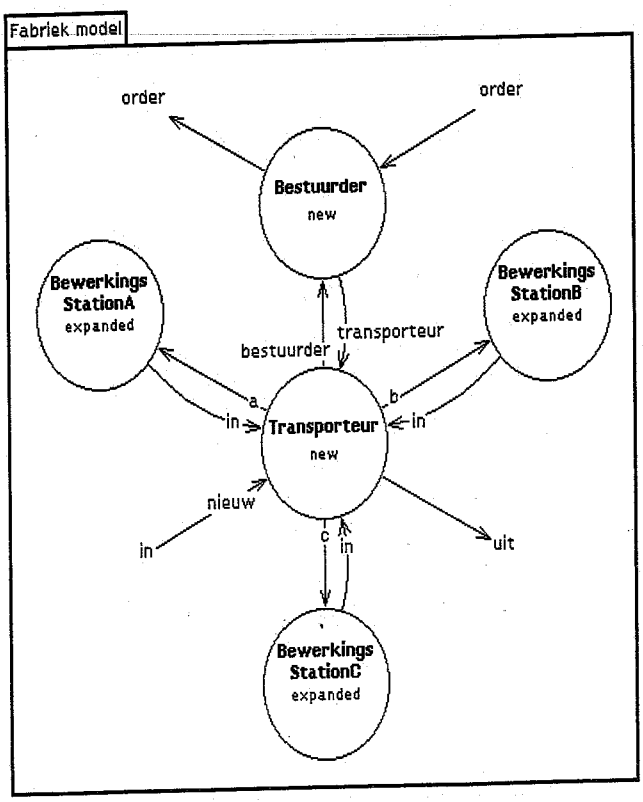


Fig. 10. Het model van de fabriek

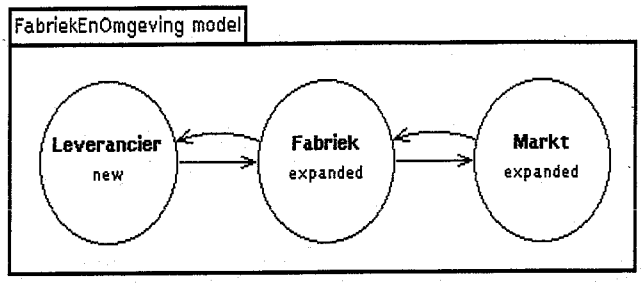


Fig. 11. Het model van de fabriek en zijn omgeving

vereist dat het systeem wordt bestuurd met behulp van extra informatiestromen: naast de waarde van de wip is andere informatie nodig om zo'n systeem te besturen, zoals welke machine werkt nu aan welk produkt, wanneer heeft een machine nieuw materiaal nodig. In het volgende voorbeeld wordt het model daarom omgebouwd tot een model met expliciete informatiestromen.

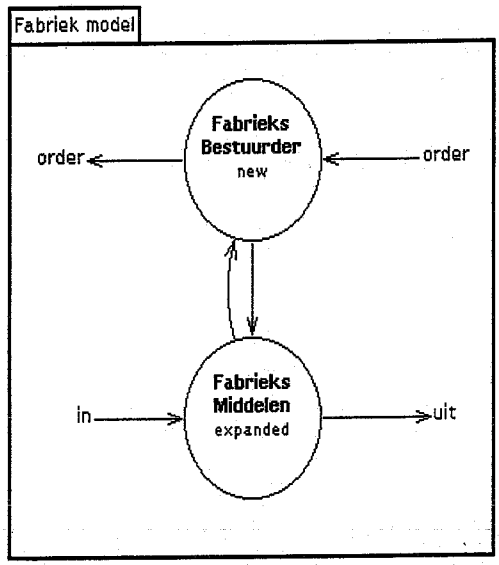


Fig. 12. Het model van de fabriek

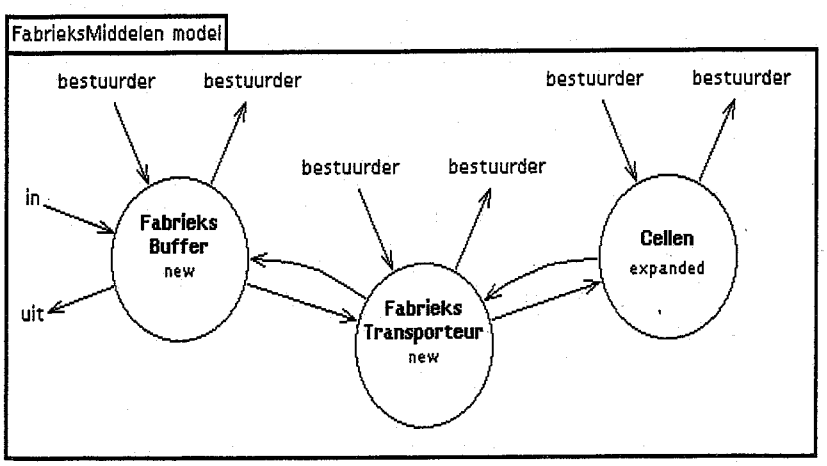


Fig. 13. Het model van de fabrieksmiddelen

Een bewerkingscentrum met informatiestromen

In dit model wordt verondersteld dat de markt het bewerkingscentrum aanstuurt. Dit betekent dat door de markt orders worden gegenereerd die door het systeem worden verwerkt. Daarnaast wordt aangenomen dat het systeem zijn wip constant houdt. Dit betekent dat de levertijd (minimaal de doorlooptijd van het systeem) ongeveer een constante is. De levertijd kan worden aangepast door de hoeveelheid onderhanden werk aan te passen. Dit zal dan weer ten koste gaan van de throughput van het systeem.

Figuur 8 toont het model van de fabriek en zijn omgeving. De leverancier, de fabriek en de afnemer zijn weergegeven door de (blad-)processor Leverancier en de geëxpandeerde processoren Fabriek en Markt.
De Markt bestaat uit een ordergenerator die de verschillende orders genereert met bijbehorende specificatie en de eigenlijke afnemer die produkten ontvangt. Dit is weergegeven in figuur 9.

De formele beschrijving van de Ordergenerator luidt:

Processor OrderGenerator
instance variables: bew1 bew2 bew3

initializeTasks
bew1 ← SampleSpace data: #(„a" „b" „c").
bew2 ← SampleSpace data: #(„a"„b"„c").
bew3 ← SampleSpace data: #(„a" „b" „c")

body
self send:
(Order
with: bew1 next
with: bew2 next
with: bew3 next)
to: „uit"

De OrderGenerator produceert een order met bijbehorende specificatie. Deze order wordt naar de Fabriek gestuurd.

De beschrijving van de Afnemer komt overeen met die van de vorige voorbeelden.

De formele beschrijving van de Leverancier luidt:

Processor Leverancier
body
signaal ← self receivefrom: „in".
self send: Materiaal new to: „uit"

De Leverancier verstuurt op verzoek van de Fabriek nieuw materiaal. De uit te voeren bewerkingsstappen worden in de Fabriek aan het materiaal toegevoegd!

In figuur 10 is de Fabriek weergegeven. Initieel zorgt de Bestuurder ervoor dat het systeem met een wip-tal nieuwe materialen wordt gevuld: hij ontvangt orders, verstuurt deze orders naar de Transporteur en geeft signalen naar de Leverancier dat nieuw materiaal dient te worden geleverd. Vervolgens accepteert de Bestuurder pas weer een order van de Markt nadat de Transporteur heeft gemeld dat er een produkt is gereedgekomen. De Bestuurder geeft de nieuwe order vervolgens door naar de Transporteur, en bestelt materiaal bij de Leverancier.

De formele beschrijving van de Bestuurder luidt:

Processor Bestuurder

```
initialActions
    | wip order |
    wip ← ?.
    wip timesRepeat:
        [order ← self receiveFrom:
        „in".
        self send: „signaal" to:
        „uit".
        self send: order to: „trans-
        porteur"]

body
    | gereed order |
    gereed ← self receiveFrom:
    „transporteur'.
    order ← self receiveFrom:
    „in".
    self send: „signaal"to: „uit".
    self send: order to: „transpor
    teur"
```

De Transporteur gebruikt de order om het materiaal te voorzien van een bewerkingsbon. De formele beschrijving van de Transporteur luidt:

Processor Transporteur

```
body
  nieuwMateriaal
  self
     receiveFrom: „order"
     then:
        [:order
        nieuwMateriaal ← self re-
        ceiveFrom: „nieuw".
        nieuwMateriaal bon: order.
        self send: nieuwMateriaal
           to: nieuwMateriaal bon
           first]
     or: „in"
     then: [:materiaal materiaal
           zijnAlleOpdrachtenAf-
           gewerkt
        ifTrue:
           [self send: materiaal to:
           „uit".
           self send: „gereed"to:
           „bestuurder"]
        ifFalse: [self send: mate-
           riaal to: materiaal bon
           first]]
```

De Transporteur probeert een order te ontvangen van de Bestuurder of materiaal te ontvangen uit één van de drie BewerkingsStations. Indien het een order betreft dan betekent dit dat nieuw materiaal via de poort „nieuw" moet worden ontvangen. De bewerkingsbon dient aan het materiaal te worden meegegeven. Het materiaal met bewerkingsbon wordt naar het betreffende BewerkingsStation verstuurd. Indien het materiaal uit een van de BewerkingsStations komt dan dient te worden onderzocht of het materiaal is afgewerkt.

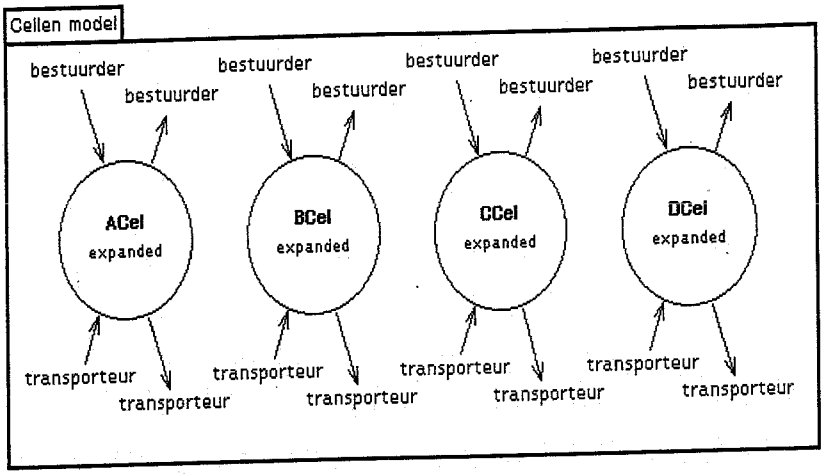


Fig. 14. Het model van de cellen

De resultaten van het oude model van het bewerkingscentrum en van het bovenstaande model zijn identiek, wanneer men de parameter wip varieert. Zo is op dezelfde manier de bedrijfskarakteristiek te bepalen. Maar het voordeel van het laatste model is groot: uitbreidingen kunnen namelijk veel gemakkelijker worden aangebracht, waardoor de praktijk-waarde van het model toeneemt. Aan de gescheiden materiaal- en de informatiestromen zijn bijvoorbeeld omsteltijden en transporttijden toe te voegen. Daarnaast vindt men in een werkelijke fabriek vaak op meerdere niveaus job-produktie. Dan is de enkelvoudige wip-besturing zoals hier gepresenteerd niet voldoende, maar kunnen hiërarchische niveaus worden toegevoegd. In een waferfabriek bijvoorbeeld

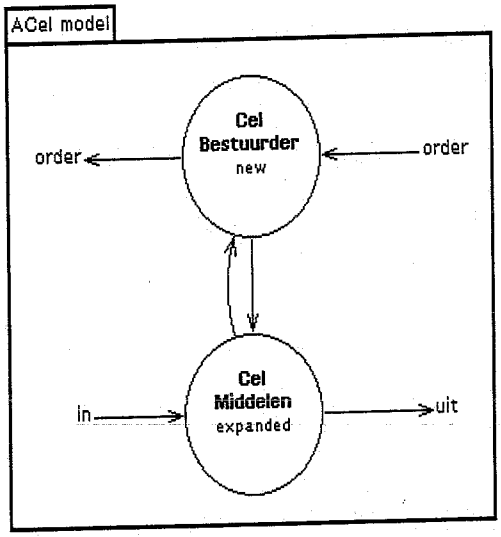


Fig. 15. Het model van deAcel

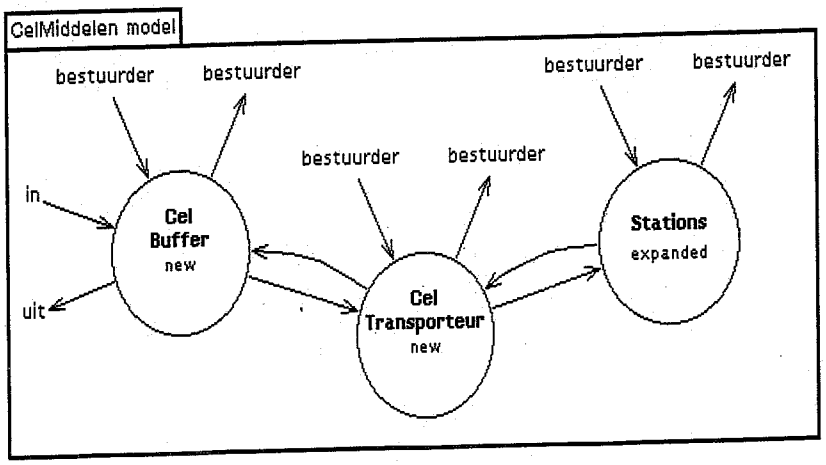
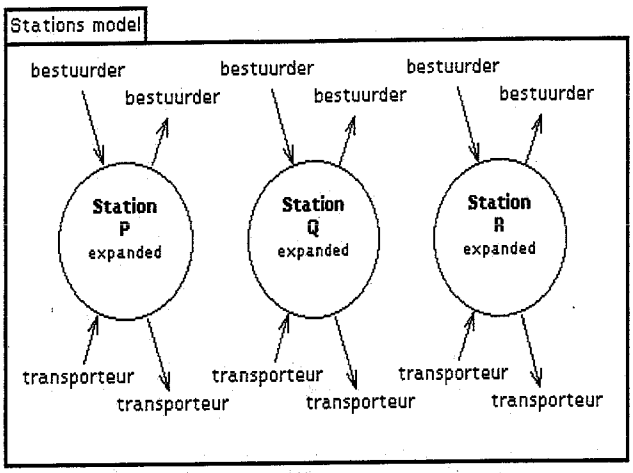


Fig. 16. Een model van een CelMiddel

Fig. 17. Het model van de stations

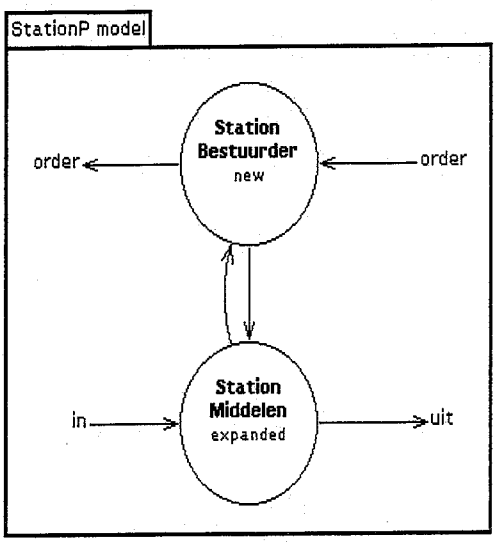


vindt men enige cellen die door de integrated circuits meerdere keren worden aangedaan. Op een volgend niveau heeft iedere cel opnieuw een job-produktie.

Een bewerkingscentrum met hiërarchische besturingen

Door Smit [1992] is een architectuur ontwikkeld voor het modelleren van job-produktie fabrieken op meerdere hiërarchische niveaus. Op ieder niveau vindt men daarin een vergelijkbare besturing. De figuren 11 tot en met 19 illustreren de verschillende hiërarchische niveaus met hun besturing.
In dit voorbeeld zijn vier niveaus opgenomen, t.w. het fabriek-, cel-, station-, en machineniveau.
Met deze architectuur is het mogelijk om zowel lokaal als globaal een fabriek te besturen. Hierbij is het dan de kunst om vast te stellen wat op ieder niveau dient te worden bestuurd. Keuzes worden meestal gemaakt vanuit een bedrijfscultuur. Binnen de ene cultuur wil de manager tot in detail het doen en laten van de manschappen controleren. Binnen een andere cultuur schetst de manager slechts de grote lijnen. In het eerste geval blijkt een weinig flexibele besturing met veel communicatie te ontstaan. In het tweede geval bestaat de mogelijkheid dat gekozen oplossingen niet optimaal zijn voor de fabriek in zijn geheel.

Fig. 18. Het model van een stationP



Nabeschouwing

In dit artikel zijn met de procescalculus een aantal modellen van job-produktie fabrieken gebouwd. Deze modellen zijn in vergelijking met de modellen van flow-produktie fabrieken complexer. Dit komt voornamelijk omdat bij job-produktie de route van de produkten kan verschillen. De gepresenteerde modellen geven aan hoe een job-produktie fabriek kan worden gemodelleerd en bestuurd. Belangrijke grootheden voor de beoordeling van de prestatie van de fabriek zijn in de modellen opgenomen: de materiaal-doorlooptijd en de materiaal throughput. Deze grootheden kunnen worden geregeld door de juiste keuze en beheersing van de wip, de hoeveelheid onderhanden werk. Het resultaat van de modellen met zo'n wip-regeling is weergegeven in de bedrijfskarakteristiek.
Gedemonstreerd is nu hoe met behulp van de procescalculus en de simulator de bedrijfskarakteristiek van een job-produktie fabriek is te bepalen. Veranderingen van het systeem; bijvoorbeeld grotere machine-capaciteiten, kortere omsteltijden en andere bewerkingspatronen zijn gemakkelijk in het model aan te brengen en met de simulator door te rekenen. Zo kan het effect van veranderingen in een complexe job-produktie fabriek worden bepaald. In een aantal praktijk-cases is deze methode inmiddels met succes toegepast.

Om fabrieks-modellen zo natuurgetrouw mogelijk te maken zijn vaak schattingen van gegevens nodig. De werkelijke gegevens zijn bijvoorbeeld niet beschikbaar, of moeilijk te verzamelen. In een volgend artikel zal nader worden ingegaan op schattingen, statistiek en de mogelijkheden die de procescalculus op dit gebied biedt.

Foto 1 toont één van de etsmachines die voorkomen in een etscel. De etscel staat in een fabriek voor de produktie van integrated circuits bij Philips Consumer IC te Nijmegen.

Deze etscel, maar ook de in de fabriek aanwezige plasmacel, zijn job-produktie georiënteerd. Met behulp van de procescalculus zijn van deze cellen modellen opgesteld, waarin de procestijden zijn verwerkt. Het gedrag van de operators is met behulp van tijdstudies vastgesteld en ook deze zijn in de modellen ingebracht. Bij validatie bleek dat de beide opgestelde modellen redelijk goed met de werkelijkheid van deze cellen overeenkwamen.
Met behulp van aanpassingen in elk model is onderzocht wat het effect is van een verandering in het gedrag van de operators. Hierbij is vooral gelet op de verbeteringen van de doorlooptijd en de throughput. Tevens is onderzocht wat het effect zou zijn indien alle handelingen in de etscel zouden worden gemechaniseerd en geautomatiseerd.
Tenslotte is ook het effect onderzocht bij toepassing van verschillende besturingen.
Deze studies werden uitgevoerd door D.G. Meeusen, W. Penning en L.A.J. Vermarien.

Opgave 1
Hoeveel produkten kunnen door het bewerkingscentrum (uit het eerste model) per uur maximaal worden bewerkt.

Opgave 2
Hef de „deadlock" in het eerste model op door een Buffer voor de Transporteur te plaatsen.

Opgave 3
Pas de beschrijving van de Bewerker in het eerste model zodanig aan dat bij twee opeenvolgende identieke opdrachten het materiaal niet naar de Transporteur wordt verstuurd, maar direct door de Bewerker wordt bewerkt.

Opgave 4
Waarom is het toegestaan om het aantal produkten te nemen als een goede maat voor de wip in plaats van de hoeveelheid onderhanden werk in uren?

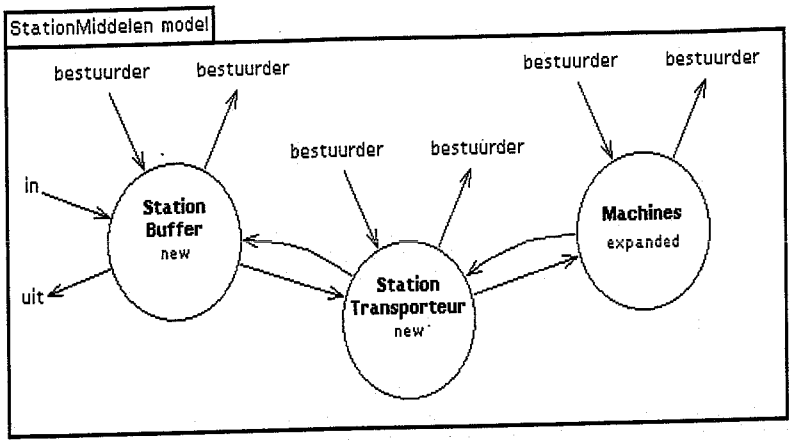


Fig. 19. Het model van de stationsmiddelen

Opgave 5
Toon aan dat figuur 6 uit figuur 7 kan worden afgeleid en dat figuur 7 uit figuur 6 kan worden afgeleid (Er geldt: gemiddelde wip = gemiddelde prestatie * gemiddelde doorlooptijd [Little, 1961, Kettner, Bechte, 1981]).

Opgave 6
Bouw een model van een bewerkingscentrum met drie bewerkingsstations en een transporteur. Op ieder bewerkingsstation kunnen twee bewerkingen worden uitgevoerd. Het omstellen van de ene naar de andere bewerking bedraagt 5 minuten. De besturing wordt uitgevoerd door een aparte besturing. Ontwerp een besturing waarbij zo weinig mogelijk hoeft te worden omgesteld. De levertijd speelt hierbij geen rol van betekenis.