TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Procescalculus bij modelleren flow-shop fabrieken

Citation for published version (APA):
Rooda, J. E., & Arentsen, J. H. A. (1991). Procescalculus bij modelleren flow-shop fabrieken. *Mechanische Technologie*, *7*(December), 10-20.

Document status and date:
Gepubliceerd: 01/01/1991

Document Version:
Uitgevers PDF, ook bekend als Version of Record

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

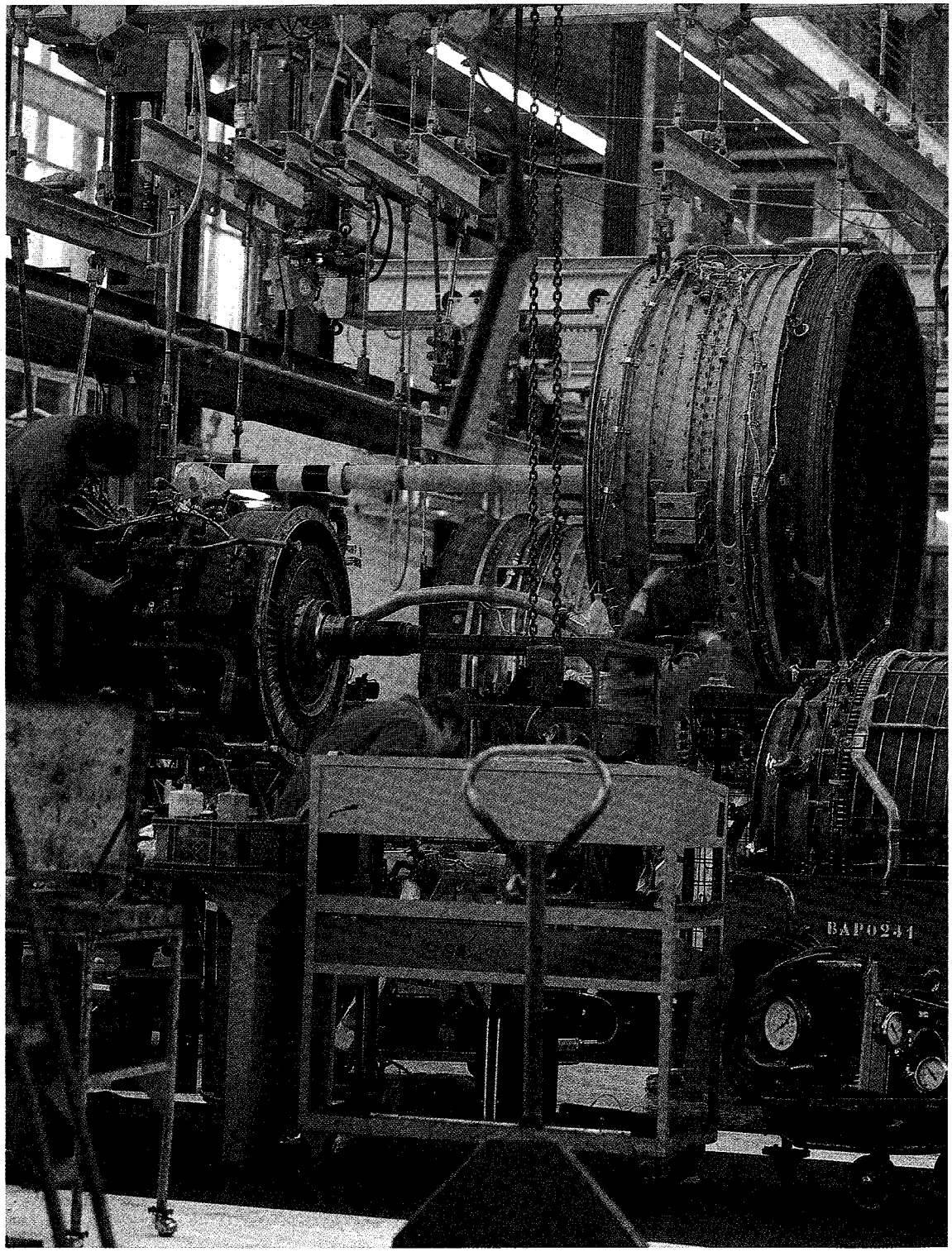
Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 04. Oct. 2023

Procescalculus bij modelleren f



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

flow-shop fabrieken

In dit artikel gebruiken de auteurs de procescalculus om een produktiesysteem te modelleren. Hierbij beperken ze zich tot fabrieken met een convergerende flow-produktie, zoals een assembleerlijn. In het eerste model wordt de assembleerlijn gebouwd, waarbij wordt verondersteld dat de leverancier alle halffabrikaten kan leveren en de afnemer alle produkten kan afnemen. Het tweede model beschrijft de assembleerlijn voorzien van buffers om de afstemming tussen de verschillende assembleurs te verbeteren. In het derde model van de assembleerlijn wordt verondersteld dat assemblage pas gebeurt als de markt om produkten vraagt. Vervolgens wordt globaal het „total factory model" gepresenteerd, waarbij zowel de materie-, de informatie- als de waarde- of geldstromen zijn aangegeven. Enige opgaven zijn toegevoegd. Dit artikel wordt afgerond met een praktijkvoorbeeld van een gemodelleerde assembleerlijn.

Om het gedrag van een industrieel systeem, zoals een fabriek, te beschrijven, kan men drie aspectsystemen onderscheiden [Rooda, Arentsen, 1983].
Het eerste aspectsysteem, het primaire systeem, wordt geassocieerd met de materiaalstromen. Het doel van dit aspectsysteem is om de produkten voort te brengen.
Het tweede aspectsysteem, het secundaire systeem, wordt geassocieerd met de informatiestromen. Het doel van dit aspectsysteem is om de voortbrenging van de produkten te besturen.
Het derde aspectsysteem, het tertiaire systeem, wordt geassocieerd met de energiestromen. Het doel van dit aspectsysteem is om het gehele systeem in stand te houden. In deze context wordt dit laatste aspectsysteem geassocieerd met waarde- of geldstromen.

Er worden in de literatuur [Buffa, Sarin, 1987] verschillende begrippen gehanteerd om industriele systemen in te delen. Een eenvoudige maar doeltreffende indeling is een onderverdeling in flow-produktie en job-produktie. (In plaats van flow-produktie en job-produktie spreekt men ook wel over produkt-georinteerde en procesgeoriënteerde produktie).In de automobielindustrie treft men meestal flow-produktie aan, terwijl een machinefabriek een voorbeeld is van een fabriek met job-produktie.
Omdat het eenvoudiger is om modellen op te stellen van fabrieken met een flow-produktie dan met een job-produktie zullen ter introductie in dit artikel met behulp van de procescalculus [Rooda, 1991] een serie eenvoudige modellen van dit type fabrieken worden gepresenteerd. Hierbij wordt een zogenaamde convergente produktielijn beschouwd: uit een aantal componenten wordt een produkt samengesteld. Een verdere beperking is dat slechts één type produkt wordt geassembleerd.

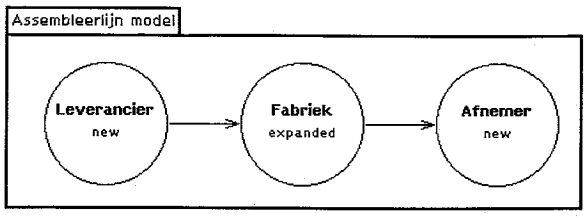


Fig.1.Het model van de assembleerlijn

In een volgend artikel zal een serie modellen van een fabriek met job-produktie aan de orde komen.

Een assembleerlijn

Bij een fabriek die produkten assembleert, wordt verondersteld dat de leverancier in staat is om de basiscomponenten te leveren en dat de afnemer alle geassembleerde produkten afneemt. Tevens wordt verondersteld dat er altijd voldoende componenten in de fabriek aanwezig zijn om te kunnen assembleren. De capaciteit van de assembleerlijn wordt bepaald door de assembleurs. De assembleerlijn zelf bestaat uit 3 in serie geschakelde assembleurs met hun bijbehorende opslag. De assembleurs werken gelijktijdig. Het assembleren vindt handmatig plaats. De assembleertijd is uniform verdeeld en ligt voor iedere werkplek tussen de 1 en 3 minuten. De lijn produceert slechts één produkttype: omstelling vindt niet plaats.

Met behulp van de procescalculus wordt een model van deze assembleerlijn opgesteld, onder meer om de capaciteitsbenutting en wachttijden vast te stellen. 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 (geexpandeerde) 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 formele beschrijving van de Leverancier luidt:

```
Processor Leverancier
body
  self send: „basisComponent"
  to: „uit"
```

De Leverancier is dus altijd in staat om de (basis-)componenten te leveren
Het model van de Fabriek is gegeven in figuur 2. Het model omvat 3 assembleerstations. In ieder assembleerstation vindt een assembleerhandeling plaats. Een assembleerstation wordt weergegeven door de geexpandeerde processor AssembleerStation. (Deze Assembleerstations zijn soortgenoten.) Een AssembleerStation, figuur 3, bestaat uit de bladprocessor Opslag en de bladprocessor Assembleur. De Opslag zorgt voor toelevering van een component. In de Assembleur vindt de assemblage plaats. De formele beschrijving van de Opslag luidt:

```
Processor Opslag
body
  self send: „component" to:
  „uit"
```

In dit model wordt verondersteld dat „basisComponenten" door een toelevancier worden aangevoerd, terwijl de fabriek zelf over voldoende „componenten" beschikt. Geen onderscheid wordt gemaakt tussen de verschillende componenten: er wordt volstaan met een enkele omschrijving voor de componenten.

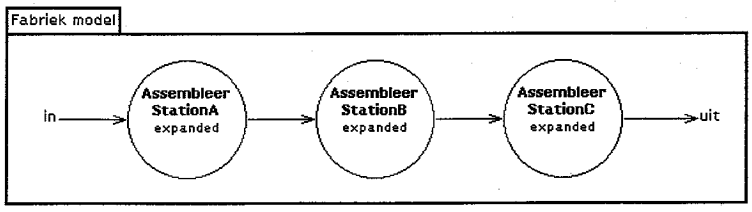


Fig.2.Het model van de fabriek

Het gedrag van een Assembleur kan worden beschreven door de volgende informele beschrijving:

```
Processor Assembleur
Voer cyclisch uit:
  probeer een basisComponent
  via de ontvang-poort „in" te
  ontvangen
  en wacht zolang dit niet lukt.
  ontvang een component via de
  ontvang-poort „opslag".
  assembleer deze twee compo-
  nenten.
  de assembleertijd ligt hierbij
  tussen de 1 en 3 minuten.
  probeer het (deels) geassem-
  bleerde produkt te verzenden
  via de zend-poort „uit"
```

De formele beschrijving van de Assembleur luidt:

```
Processor Assembleur
instance variable: assembleerTijd

initializeTasks
  assembleerTijd ^ Uniform
  from: 1 minutes to: 3 minutes

body
  | basisComponent compo-
  nent produkt |
  basisComponent ^ self recei-
  veFrom: „in".
  component ^ self receiveFr-
  om: „opslag".
  produkt ^ self assembleer: ba-
  sisComponent
    en: component
    gedurende: assembleerTijd
    next.
  self send: produkt to: „uit"
```

Met behulp van een kansverdeling wordt initieel de assembleertijd vastgelegd. Deze ligt tussen de 1 en 3 minuten en is uniform verdeeld (Iedere Assembleur heeft zijn eigen kansverdeling). De Assembleur probeert eerst de basisComponent en dan de component te ontvangen. De opdracht „self assembleer: en: gedurende:" zorgt voor het assembleren van de twee componenten, waarbij de assembleerTijd wordt vastgesteld door een trekking uit de uniforme kansverdeling door de opdracht „assembleerTijd next". Een produkt ontstaat als resultaat van deze opdracht. Hoe het assembleerproces precies verloopt, wordt hier

verder niet beschreven. Deze opdracht bevat in ieder geval de opdracht „self workDuring:". Het (deels) geassembleerde produkt wordt vervolgens via „uit" verzonden.

Met de beschrijvingen van de processoren Leverancier, Afnemer, Opslag en Assembleur is de assembleerlijn met zijn omgeving volledig beschreven. Zoals in voorgaande artikelen is beschreven kan nu met behulp van de simulator het gedrag van dit model worden onderzocht. Dan blijkt dat deze assembleerlijn per 12 uur 290 produkten (ongeveer 24 produkten per uur) kan afleveren. Bovendien blijkt dat AssembleurB ongeveer 10% van zijn tijd heeft moeten wachten op een (deels) geassembleerd produkt en dat deze ongeveer 10% van zijn tijd heeft moeten wachten omdat deze het produkt niet heeft kunnen verzenden, zie figuur 4. De prestatie van de assembleerlijn kan worden verbeterd door de verschillen in assembleertijd op te vangen door de introductie van buffers. Het volgende voorbeeld toont deze verbetering.

Een gebufferde assembleerlijn
Het model van figuur 3 wordt nu uitgebreid met buffers. De verwachting is dat deze buffers zorgen voor een betere afstemming van de verschillende assembleerprocessen. In figuur 5 wordt het nieuwe model van het AssembleerStation getoond. Het AssembleerStation bestaat uit de Opslag, de Assembleur en een Buffer. De Opslag en de Assembleur zijn identiek aan die uit het vorige model. Strikt genomen is de Buffer in AssembleerStationC overbodig.
Een informele beschrijving van de buffer is de volgende:

Processor Buffer
Voer cyclisch uit:
  probeer de component, die het langst
    in de buffer aanwezig is te verzenden
  of
  probeer een component te ontvangen
    en voeg deze aan de buffer toe

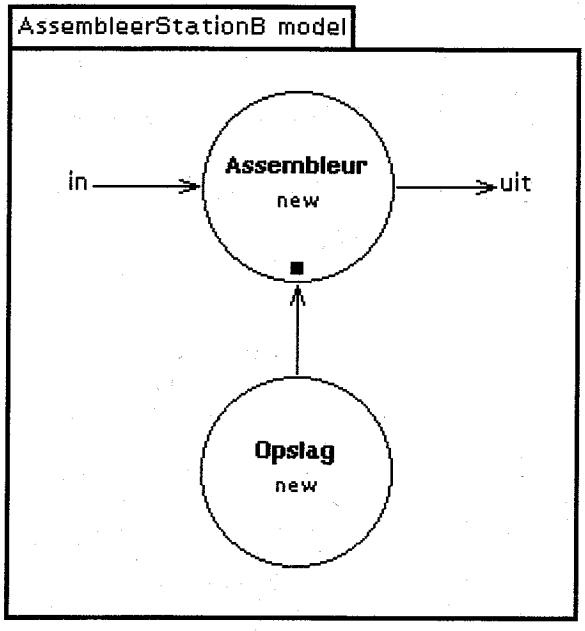
Voordat de formele beschrijving van de Buffer wordt gegeven, zal eerst een beschrijving worden gegeven van het begrip geordende verzameling. De geordende verzameling zal straks worden gebruikt voor het formaliseren van het model van de Buffer. Een geordende verzameling („OrderedCollection") is een object waarin andere objecten kunnen worden bewaard. Aan een geordende verzameling kan worden gevraagd of deze objecten bevat; een object kan aan de geordende verzameling worden toegevoegd; een object kan uit een geordende verzameling worden verwijderd. De informele omschrijving van een geordende verzameling („OrderedCollection") luidt:

Object OrderedCollection
isEmpty
  levert true indien de verzameling leeg is
  levert false indien de verzameling niet leeg is

addLast: eenObject
  voegt eenObject toe als laatste in de verzameling

removeFirst
  levert het eerste object uit de verzameling.
  dit object wordt uit de verzameling verwijderd.

Een geordende verzameling kan worden gemaakt door de opdracht „OrderedCollection new". De volgende tekst geeft de opdrachten weer voor het creëren van een geordende verzameling,



het plaatsen van enige objecten in de geordende verzameling en het verwijderen van een object uit de geordende verzameling:

```
lijst ^ OrderedCollection new.
lijst addLast: „aap".
lijst addLast: „noot".
lijst addLast: „mies".
aantal ^ lijst size.
  „dit levert 3 op"
object ^ lijst removeFirst.
  „object wordt „aap",
```

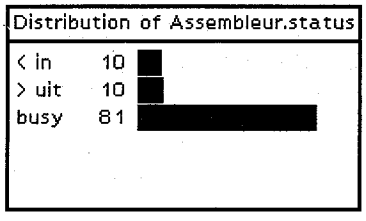


Fig.4.De verdeling van assembleerstation B

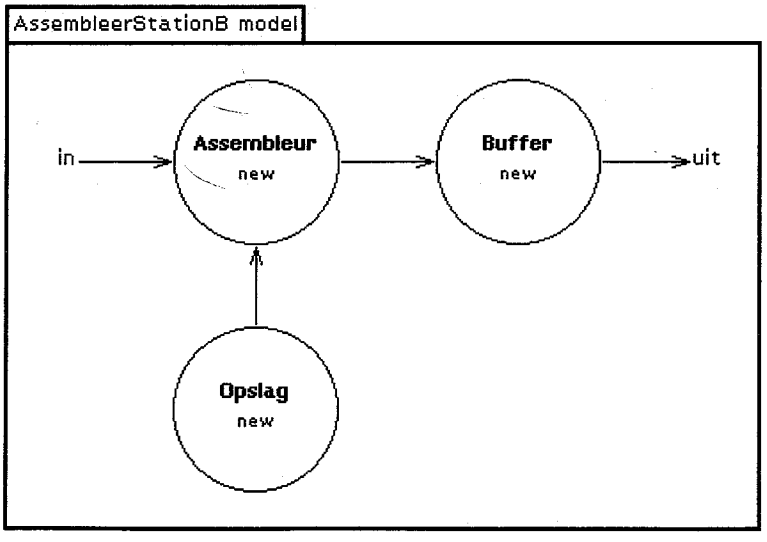
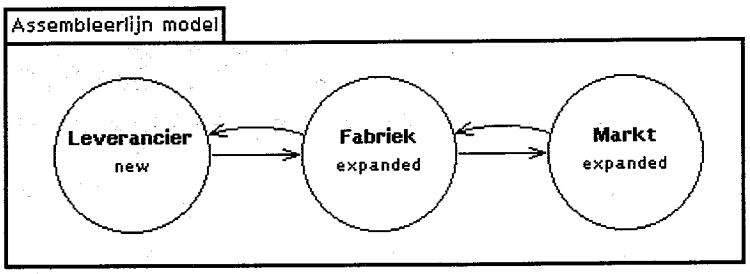


Fig.5.Het model van assembleerstation B

Fig.6.Het model van de assembleerlijn



„aap" wordt uit de verzameling verwijderd"
aantal ^ lijst size
„dit levert 2 op"

Voor het modelleren van industriele systemen wordt vaak gebruik gemaakt van geordende verzamelingen. Een geordende verzameling is vergelijkbaar met een lijst: men schrijft bij aan de onderkant en men streept af aan de bovenkant.

Met behulp van de beschrijving van de geordende verzameling wordt de formele omschrijving van de Buffer:
Processor Buffer
instance variable: lijst

initializeTasks
lijst ^ OrderedCollection new

Fig.7.Het model van de markt

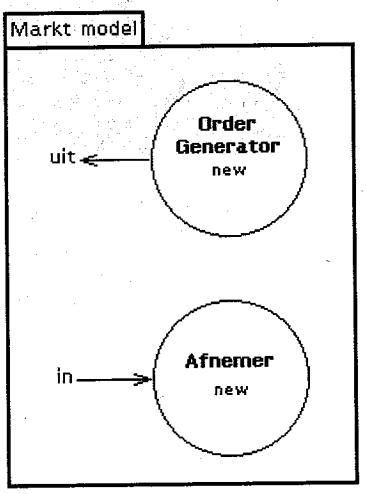
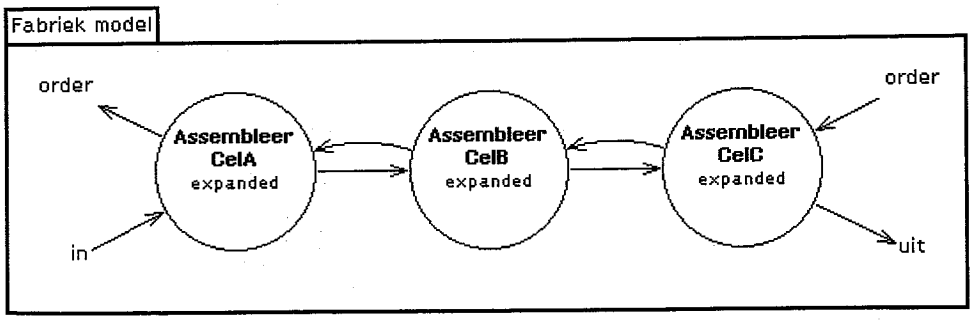


Fig.8.Het model van de fabriek



body
lijst isEmpty
ifTrue: [lijst addLast: (self receiveFrom: „in")].
self
send: lijst first
to: „uit"
then: [lijst removeFirst]
orReceiveFrom: „in"
then: [:object | lijst addLast: object]

Initieel wordt een betekenis aan „lijst" toegekend door de opdracht „lijst OrderedCollection new". Allereerst wordt onderzocht of de „lijst" in de Buffer leeg is. Is dit het geval dan wordt net zolang gewacht tot een object kan worden gelezen via de ontvang-poort „in", waarna dit object aan de „lijst" wordt toegevoegd. Dan is het mogelijk dat òf een nieuw object kan worden ontvangen òf dat een object uit de „lijst" kan worden verzonden. Indien het lukt om een object te verzenden, dan wordt na verzending het object uit de „lijst" verwijderd. Indien het lukt om een object te ontvangen dan wordt dit object aan de „lijst" toegevoegd. Het ontvangen object wordt in de opdracht „[:object | lijst addLast: object]" tijdelijk vastgehouden door de variabele „object" die zich bevindt tussen de dubbele punt en de rechte streep. De opdracht „send: to: then: orReceiveFrom: then:" stelt de modelleur in staat om zend- en ontvangakties te combineren. De gemodelleerde buffer is een zogenaamde „first-in first-out" buffer.

Met behulp van de simulator blijkt dat na enige tijd de assembleerlijn circa 30 produkten per uur af kan leveren. De introductie van buffers heeft de capaciteit van de lijn dus inderdaad verhoogd met circa 25%.
Na een experiment waarbij de assembleerlijn 40 uur heeft gewerkt, en waarbij circa 1200 produkten zijn geassembleerd, blijkt dat in de Buffer behorende bij AssembleerStationA minimaal 0, maximaal 14 en gemiddeld 3,7 (deels geassembleerde) produkten aanwezig zijn. Voor de Buffer in AssembleerStationB wordt gevonden respectievelijk 0; 20; 5,1. Zo lijkt op het eerste gezicht een buffer met 5 posities voldoende. Vervolging van het experiment levert echter een geheel ander beeld (tabel 1):

Totaal zijn in 360 uur 10748 produkten geassembleerd (in plaats van (360 * 30) – 3 = 10797).
De verklaring van deze uitkomsten is als volgt. Door het gebruik van kansverdelingen is het mogelijk dat de gemiddelde assembleertijd van AssembleurA en AssembleurB een klein beetje verschillen. Wanneer AssembleurB 1% langzamer werkt dan AssembleurA dan heeft dit tot gevolg dat BufferA 12 (deels) geassembleerde produkten meer bevat in 40 uur. In het interval 40 – 80 heeft AssembleurC te langzaam geassembleerd. In het interval 160 – 200 heeft AssembleurB te langzaam geassembleerd. De veranderingen in de vereiste buffercapaciteit kunnen worden voorkomen door de buffers een beperkte capaciteit te geven (bijvoorbeeld 5), dan wel AssembleurA iets langzamer te laten assembleren dan AssembleurB (bijvoorbeeld 0,5%) en AssembleurB iets langzamer te laten assembleren dan AssembleurC (bijvoorbeeld 0,5%). In beide gevallen zal dit een iets kleinere prestatie tot gevolg hebben.

Een assembleerlijn met besturing op order

Interval (uren)	BufferA Min	Max	Gem	BufferB Min	Max	Gem
0 – 40	0	14	3,7	0	20	5,1
40 – 80	0	12	4,4	10	28	19,0
80 – 120	0	20	11,7	19	37	26,6
120 – 160	5	32	19,8	11	34	24,1
160 – 200	6	37	24,6	0	22	6,6
200 – 240	18	37	27,5	0	12	5,0
240 – 280	20	43	32,4	0	15	6,9
280 – 320	6	41	21,2	0	15	6,0
320 – 360	10	42	30,7	0	15	5,1

Tabel 1. Statistiek van BufferA en BufferB

Tot dusver zijn modellen opgesteld van assembleerlijnen waarbij er vanuit is gegaan dat de geassembleerde produkten altijd worden afgenomen. Met andere woorden de fabriek „drukt" de produkten naar de afnemer. Vaak is de situatie echter anders: de afnemer „trekt" de produkten uit de fabriek. Er worden dan alleen produkten geassembleerd indien de afnemer erom vraagt. Een „trek"-fabriek kan worden gemodelleerd door in het beschreven model de buffers een beperkte capaciteit te geven: de buffers lopen vol en het assembleren stopt, indien de markt niets afneemt. Deze oplossing, die boven reeds geschetst is om het fluctuerende gedrag in de buffers op te vangen, heeft het nadeel dat de fabriek vol ligt met (deels geassembleerde) produkten. Een betere oplossing is om de produktie te besturen door een informatiestroom.

De meest eenvoudige door informatiestromen bestuurde fabriek is een fabriek waarbij op order wordt geproduceerd. Met andere woorden na ontvangst van een order wordt met de produktie een begin gemaakt. Bij een dergelijk systeem verstuurt de afnemer orders naar de fabriek: informatiestromen gaan hun intrede doen. Het is hierbij wel noodzakelijk dat de order gereed is vóór het verstrijken van de leverdatum van die order.
Met behulp van de procescalculus wordt een model opgesteld van deze fabriek. Figuur 6 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 geexpandeerde processoren Fabriek en Markt.
De Markt bestaat uit een ordergenerator die de verschillende orders genereert en de eigenlijke afnemer die de produkten ontvangt. Deze worden gemodelleerd in figuur 7 door de (blad-) processoren OrderGenerator en Afnemer.
Er wordt aangenomen dat 20 produkten per uur worden afgenomen, met andere woorden de OrderGenerator dient orders voor deze hoeveelheid te versturen.
De formele beschrijving van de OrderGenerator luidt:

Processor OrderGenerator
body
self workDuring: 3 minutes.
self send: „order" to: „uit"

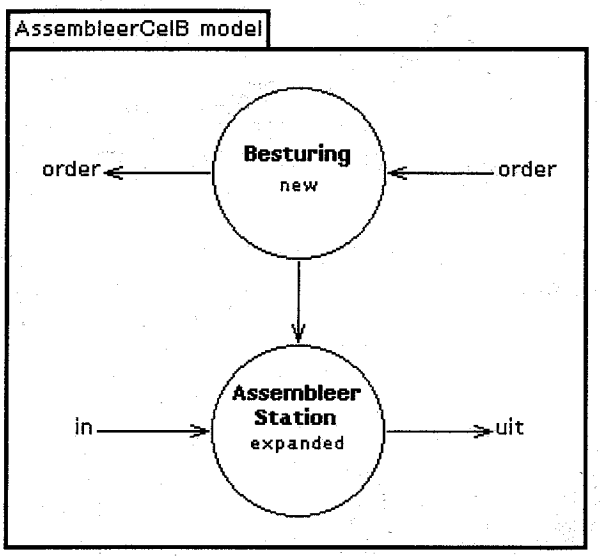


Fig.9.Het model van assembleercel B

Omdat slechts één type produkt wordt gevraagd is het niet noodzakelijk om deze order nader te specificeren: de order wordt weergegeven door de tekst „order".
De beschrijving van de Afnemer komt overeen met die van de vorige voorbeelden.

De formele beschrijving van de Leverancier luidt:

Processor Leverancier
body
order self receiveFrom: „in".
self send: „component" to: „uit"

Met andere woorden de Leveran-

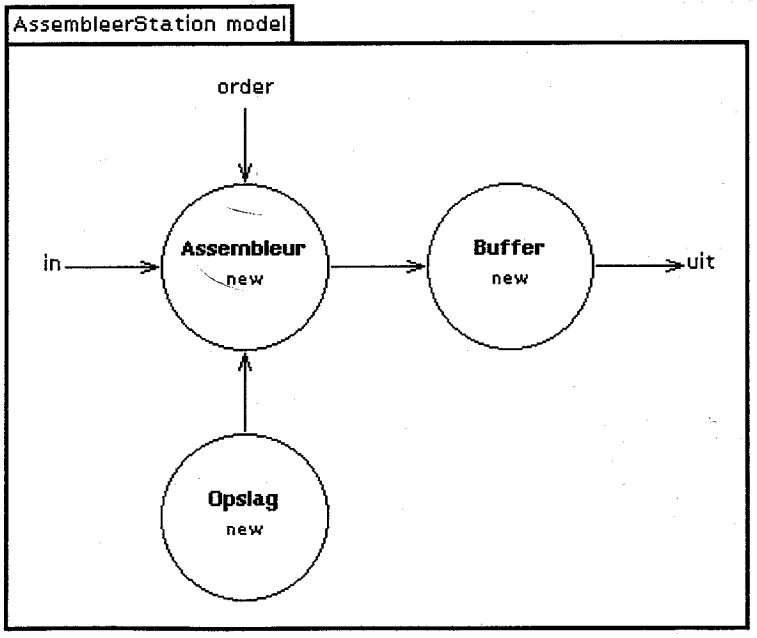


Fig.10.Het model van het assembleerstation

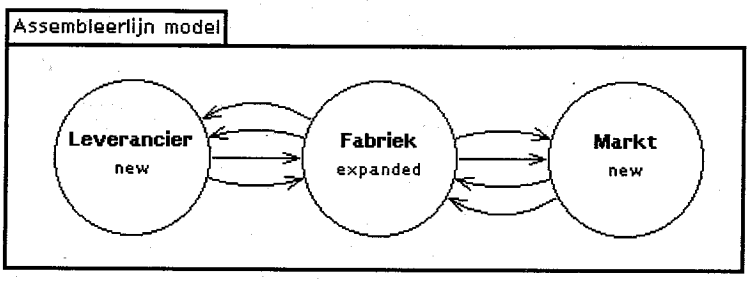


Fig.11.Het model van de assembleerlijn

cier verstuurt een component naar de Fabriek, indien hij een opdracht tot levering heeft ontvangen.
De Fabriek is weergegeven in figuur 8. De assembleercellen zijn met elkaar gekoppeld via materie- en informatiestromen. De AssembleerCel bestaat uit een Bestuurder en een AssembleerStation. Dit is weergegeven in figuur 9. De Bestuurder ontvangt een order tot levering van een produkt. De Bestuurder plaatst een copy van de order tot levering van componenten bij de vorige AssembleerCel of de Leverancier. Daarnaast geeft de Bestuurder opdracht aan het AssembleerStation om de componenten te assembleren. Dit betekent dat de order zowel naar de vorige processtap wordt doorgeleid als naar het AssembleerStation wordt gezonden. Tevens mag de Bestuurder niet worden geblokkeerd in zijn akties: op die manier zou de ontvangst en doorleiding van nieuwe orders worden opgehouden. De informele beschrijving van de Bestuurder luidt:

Processor Bestuurder
body
verzend een order direct naar het AssembleerStation
of
probeer een order te ontvangen
en plaats een copie van deze order bij de voorganger
en voeg deze order toe aan de orderlijst.

De Bestuurder is te beschouwen als een doorgeefluik voor orders naar zijn voorganger en als een buffer met orders voor zijn AssembleerStation.
De formele beschrijving van de Bestuurder luidt:

Processor Bestuurder
instance variable: orderLijst

initializeTasks
orderLijst ^ OrderedCollection new

```
body
  | order |
  orderLijst isEmpty
    ifTrue:
      [order ^ self receiveFrom:
      „in".
      self send: order copy to:
      „uit".
      orderLijst add: order].

self
  send: orderLijst first
  to: „station"
  then: [orderLijst removeFirst]
  orReceiveFrom: „in"
  then:
    [ :order |
    self send: order copy to:
    „uit".
    orderLijst add: order]
```

Het AssembleerStation bestaat uit een Assembleur, een Opslag en een Buffer, figuur 10. Deze Assembleur verschilt met de vorige Assembleurs in die zin dat deze Assembleur met zijn werkzaamheden begint nadat hij een opdracht van de Bestuurder heeft ontvangen.
De formele beschrijving van de Assembleur luidt:

Processor Assembleur
body

```
  | order basisComponent component produkt |
  order ^ self receiveFrom: „bestuurder".
  basisComponent ^ self receiveFrom: „in".
  component ^ self receiveFrom: „opslag".
  produkt ^ self assembleer: basisComponent
    en: component
    gedurende: assembleerTijd
    next.
  self send: produkt to: „uit"
```

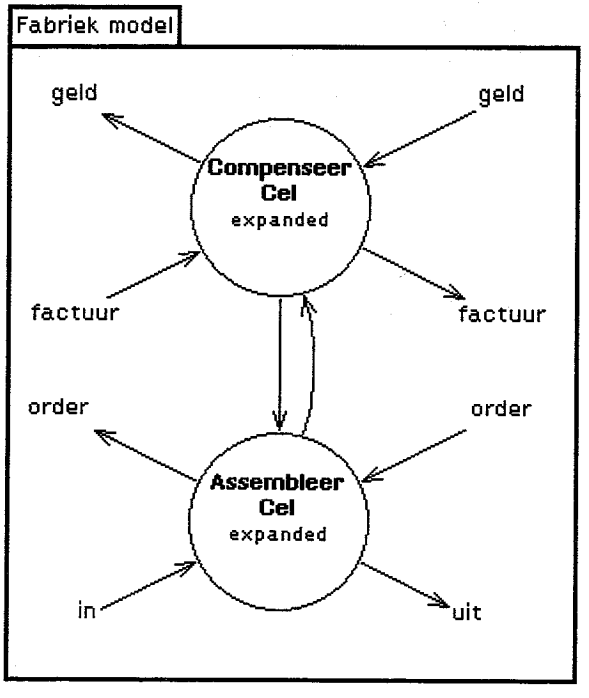
Omdat alle componenten perfect worden geassembleerd is van een open-lus besturing uitgegaan. Dit is een vereenvoudiging ten opzichte van de gesloten-lus besturingen. In dat geval wordt het resultaat van de bewerking wel aan de Bestuurder teruggemeld. De Bestuurder kan dan de bijbehorende aktie ondernemen.

De Bestuurder en het AssembleerStation zijn gemodelleerd als twee aparte processoren: gedurende de assembleertijd van het AssembleerStation kunnen de orders door de Bestuurder worden doorgegeven.

Met behulp van de simulator is vast te stellen dat per uur inderdaad 20 produkten worden geassembleerd. Omdat dit systeem niet volledig wordt belast, blijven de bufferinhouden beperkt.

In dit nog steeds eenvoudige model is er vanuit gegaan dat alle produkten op order kunnen worden geassembleerd. In de praktijk is dit vaak niet het geval. In zulke situaties worden meestal een aantal (deels) geassembleerde produkten op voorraad geproduceerd, terwijl deze (deels) geas-

Fig.12.Het model van (een enkele stap in) de fabriek



sembleerde produkten volledig worden geassembleerd nadat een order is ontvangen. Er ontstaat een zogenaamd ontkoppelpunt. Het ontkoppelpunt is dan die buffer die zich bevindt tussen het op voorraad werkende deel en het op order werkende deel van de assemblagelijn. In dit artikel wordt hier verder niet op ingegegaan. Duidelijk zal zijn dat ook deze typen van assemblagelijnen met behulp van de procescalculus kunnen worden beschreven.

Een fabriek met materie-, informatie- en geldstromen
Voor het instandhouden van een fabriek is het noodzakelijk dat er afrekening van de gefabriceerde produkten plaatsvindt. Ter illustratie is in figuur 11 het model van zo'n fabriek en zijn omgeving gegeven. In de praktijk betekent dit dat een factuur wordt verzonden naar de afnemer. De afnemer voldoet deze factuur door betaling. Met het verkregen geld is de fabrikant op zijn beurt in staat om zijn leveranciers te betalen. Een voorbeeld van zo'n fabriek is gegeven in figuur 12. Zoals eerder is geïllustreerd hoeft de orderontvangst en de orderverwerking niet synchroon te verlopen met de eigenlijke produktie. Reden om zowel de assembleerprocessen als de besturingsprocessen weer te geven door aparte processoren. Op een zelfde wijze hoeft de facturering niet samen te vallen met de ontvangst van de betalingen. De figuren 13 en 14 geven in meer detail deze aparte processoren weer.

Door Arentsen [1989] zijn de bouwstenen, zoals geschetst in de figuren 12, 13 en 14, beschreven. Dit „total factory model" biedt een architectuur voor het modelleren van de fabricage zelf en de bijbehorende besturingen. Hierbij kan men denken aan besturingen die op order, op voorraad of op mengvormen functioneren, zoals bijvoorbeeld Kanban-besturingen [Kimura, Terada, 1981; Shingo, 1981], MRP-besturingen [Orlicky, 1975; Browne, Harnen, Shivnan, 1988] en PostAnte- en AntePost-besturingen [Arentsen, 1989].
Met behulp van de procescalculus is het nu dus mogelijk om modellen van fabrieken op te stellen waarbij zowel de materiestromen met bijbehorende (order-)informatiestromen, als de geldstromen met bijbehorende (factuur-)informatiestromen te modelleren.

Op dit moment wordt aan de Technische Universiteit Eindhoven bij de leerstoelen Automatisering van de Produktie en Technische Bedrijfsvoering, onderzoek verricht om de processoren die behoren bij de geld- en factuurstromen nader te preciseren.

Nabeschouwing

De verschillende ontwikkelde modellen illustreren hoe assembleerlijnen kunnen worden ontworpen. De overgangen tussen de gepresenteerde modellen tonen enigszins hoe de ontwikkeling van de assemblage door de tijden heen is verlopen: in het begin was er alleen sprake van materiaalvoortbrenging, gevolgd door een periode waarbij ook de informatie in beschouwing wordt genomen. Nog relatief nieuw is dat men zich realiseert dat daarnaast de verschillende financiele systemen nodig zijn voor de besturing van fabrieken.

Door gebruik te maken van de procescalculus kan men een model van een nieuw te bouwen of een bestaande fabriek ontwerpen. Met behulp van de in het gereedschap aanwezige simulator is het vervolgens mogelijk om het gedrag van de verschillende samenwerkende processoren (dynamisch) vast te stellen. Aan de hand van het te leveren produktenpakket en de bijbehorende levertijden is het dan mogelijk om tijdens de ontwerpfase uitspraken te doen over bijvoorbeeld het vereiste fabricagesysteem, het gewenste besturingssysteem, het effect van een bepaalde orderverwerkingsstrategie, de invloed van seriegroottes, de invloed van seizoenen, de gevoeligheid voor omsteltijden, de te verwachten doorlooptijden en de hoeveelheid onderhanden werk.

De getoonde werkwijze toont essentiële verschillen en voordelen

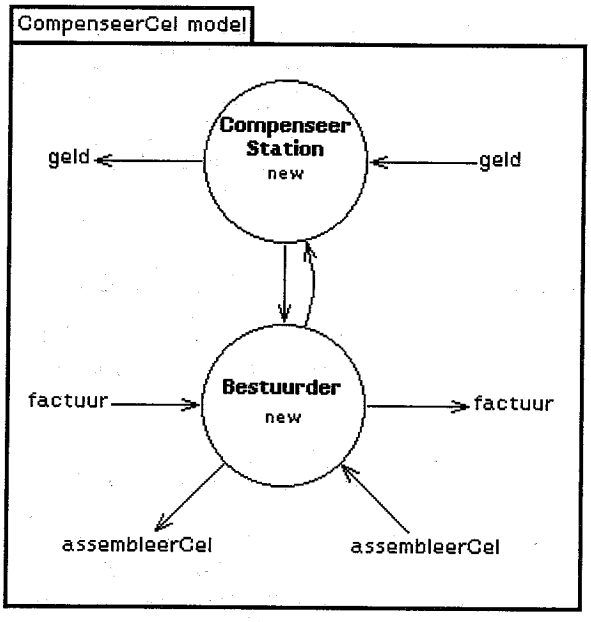
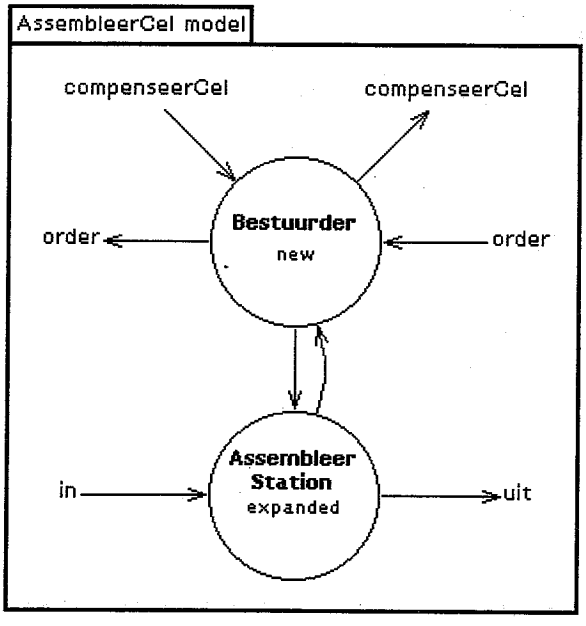


Fig.13.Het model van de compenseercel

met de werkwijze zoals die nog veel in de industrie wordt toegepast. In de industrie ontwerpt men statisch een fabriek op empirische gronden, waarna men tijdens de gebruiksfase vervolgens via de „learning curve" of via „trial-and-error" probeert een dergelijke fabriek te optimaliseren. Het voordeel van de hier getoonde werkwijze is dat men vooraf snel een volledig inzicht verkrijgt in de kwaliteit van het fabricageproces en zijn bijbehorende besturing. in kader

Foto 1 toont een lijn voor het assembleren van papiertoevoereenheden bij Stork Nolte EMI bv te Eindhoven zoals deze wor-

Fig.14.Het model van de assembleercel



den toegeleverd ten behoeve van copieermachines.
Aan de hand van processchema's is met behulp van de procescalculus een model van deze lijn opgesteld in termen van processoren en interactiepaden. Het totaal aantal bladprocessoren in het model bedraagt ongeveer 30. Door uitvoering van multi-moment opnames zijn de verschillende tijdsbestedingen van de (menselijke) assembleurs vastgesteld. Daarnaast zijn de tijden van de verschillende montagehandelingen bepaald. Deze tijdsbestedingen en tijden zijn gebruikt voor de specificatie van de verschillende bladprocessoren. Het bleek met behulp van de simulator dat het model een goede overeenkomst vertoonde met de werkelijkheid. Met dit model zijn onder andere de knelpunten in de assemblagelijn bepaald.
Deze studie werd uitgevoerd door A.H.M. Aerts.

Literatuur

Arentsen J.H.A., 1989,
Factory Control Architecture, A systems approach,
Dissertatie,
Technische Universiteit Eindhoven.

Browne J., Harhen J., Shivnan J., 1988,
Production management systems,
Addison-Wesley Publ. Comp., Workingham, UK.

Buffa E.S., Sarin R.K., 1987,
Modern Production/Operations Management,
John Wiley & Sons, London, UK.

Kimura O., Terada H., 1981,
Design and analysis of pull system, a method of multi-stage production control,
Int. J. of Prod. Research, 19 (3), 241 - 253.

Orlicky J., 1975,
Material Requirements Planning,
McGraw-Hill Inc., New York.

Rooda J.E., Arentsen J.H.A., 1983,
Een structuurmodel voor de beschrijving van transport- en produktiesystemen,
Transport + Opslag 7 (10), 88-90.

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

Rooda J.E., 1991b,
Procescalculus,
Systemen, modellen en geschiedenis,
I2 Werktuigbouwkunde 7 (8), 36-39.

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

Shingo S., 1981,
Study of Toyota production system,

Japan Management Association, Tokyo, Japan.

Opgaven

Opgave 1
Waarom is de buffer in AssembleerStationC strikt genomen overbodig in het model van de gebufferde assembleerlijn?

Opgave 2
Uit hoeveel bladprocessoren bestaat het model van de gebufferde assembleerlijn?

Opgave 3
Modelleer een „first-in first-out" buffer met een beperkte capaciteit. Hints: voeg de instance variable „capaciteit" toe. Geef „capaciteit" een waarde, bijvoorbeeld 4, in initialize-Tasks. Indien de buffer leeg is, dan kunnen alleen objecten worden ontvangen. Bepaal met behulp van de opdracht „lijst size" het aantal objecten in de lijst. Indien de buffer vol is, dan kunnen alleen objecten worden verzonden. Indien de buffer gevuld is (niet leeg en niet vol) dan kunnen objecten zowel worden verzonden als ontvangen.

Opgave 4
Waarom is de orderontvangst strikt genomen niet noodzakelijk voor de Leverancier bij het model van een assembleerlijn met besturing op order?

Opgave 5
Bouw een model van een assembleerlijn waarbij twee typen produkten worden gevraagd door een afnemer. De gewenste hoeveelheden van type1 en type2 bedragen respectivelijk 17 en 9 produkten per uur. De twee produkten zijn identiek afgezien van de component die door AssembleurB wordt geassembleerd. Voorzie AssembleurB derhalve van twee opslagprocessoren. De omsteltijd van type1 naar type2 bedraagt 1 minuut. De omsteltijd van type2 naar type1 bedraagt 0,5 minuten. AssembleurB is de flessehals van de assembleerlijn.

Opgave 6
Ontwerp tevens een strategie behorende bij opgave 5 om AssembleurB optimaal te laten produceren.