Een grafische methode voor het weergeven van parallelle processen en hun interacties binnen de proces-interactie-omgeving

Janssen, S.A.

Award date:
1990

Een grafische methode voor het weergeven van parallelle processen en hun interacties binnen de proces-interactie-omgeving

S.A. Janssen
juni 1990
Eindstudieverslag
WPA 0910

TECHNISCHE UNIVERSITEIT EINDHOVEN
Faculteit Werktuigbouwkunde
Vakgroep WPA

18 december 1989

Eindstudie-opdracht : S.A. Janssen

Afstudeerhoogleraar : Prof.dr.ir. J.E. Rooda

Begeleider : Ir. D.A. van Beek

Onderwerp : Interactieve hulpmiddelen om parallelle en sequentiële acties in een model grafisch te kunnen weergeven.

Toelichting

Bij Vredestein in Doetinchem worden besturingsprogramma's ontwikkeld met behulp van de proces-interactie benadering. De programmalistings zijn echter moeilijk leesbaar voor storingsmonteurs. Er is behoefte aan faciliteiten om betere documentatie bij de besturingsprogramma's te kunnen genereren.
Een probleem is dat er vrij veel parallellisme optreedt in de vorm van kortdurende parallelle acties, zoals het gelijktijdig aansturen van twee cylinders. Het uitbollen van dit soort parallelle acties geeft geen verbetering van de leesbaarheid van de besturing.

Opdracht

Voeg hulpmiddelen toe aan de proces-interactie omgeving waarmee besturingsprogramma's zodanig kunnen worden ontwikkeld dat parallelle en sequentiële acties in één window grafisch kunnen worden weergegeven. Hierbij moeten ook de interactiepaden zichtbaar zijn.

Verslag, etc.

Bij de secretaresse is verkrijgbaar:

1. Het memorandum "Afstuderen in de Produktietechnologie en -Automatisering".
2. "Wat moet waar en hoe in het verslag" door drs. P. Westendorp.

Prof.dr.ir. J.E. Rooda

Ir. D.A. van Beek

SAMENVATTING

Binnen de vakgroep WPA (Produktietechnologie en Automatisering) van de faculteit Werktuigbouwkunde wordt op het ogenblik onder meer onderzoek verricht naar het simuleren en besturen van machines en fabrieken.
In het kader van dit onderzoek is een programmeeromgeving ontwikkeld met behulp van de programmeertaal Smalltalk-80, waarin de in deze machines en fabrieken voorkomende parallelle processen en hun onderlinge interacties eenvoudig kunnen worden gemodelleerd. Deze programmeeromgeving wordt dan ook de proces-interactie-omgeving genoemd.
Binnen de proces-interactie-omgeving worden parallelle processen en hun onderlinge interacties grafisch weergegeven door bollen met daartussen pijlen. Deze grafische weergave is heel duidelijk zolang men alleen is geinteresseerd in de globale opbouw van een model en zolang men alleen wil weten welke processen er zijn en in welke mate het procesverloop in de verschillende bollen samenhangt.
Wil men echter meer weten dan zal men van ieder proces (bol) de procesbeschrijving moeten opvragen, waarin precies staat beschreven wat het proces doet en wanneer welke interacties plaatsvinden. Zodra het model dat moet worden gesimuleerd of bestuurd echter ingewikkelder wordt, kan men zelfs uit deze procesbeschrijvingen niet meteen alle informatie halen.
Uit deze procesbeschrijving haalt men namelijk alleen alle informatie betreffende dat ene proces, men kan aan die procesbeschrijving niet zien, wat er intussen in de andere processen gebeurt. Om deze informatie te verkrijgen, moet men de verschillende procesbeschrijvingen met elkaar vergelijken. Bij meerdere pocessen met langere procesbeschrijvingen, waarin veel interacties zijn gespecificeerd, verliest men gauw het overzicht.
Daarbij is het in de huidige grafische weergave moeilijk om voorwaardelijke structuren meteen te herkennen. Als een proces pas start zodra aan een bepaalde voorwaarde is voldaan, dan zal dit uit het bollen-en-pijlen-schema niet blijken. Men kan daarin ook niet zien, dat bepaalde processen nooit tegelijkertijd kunnen verlopen. Het ene proces loopt misschien alleen zodra aan een bepaalde voorwaarde is voldaan, terwijl het andere proces juist alleen loopt als niet aan deze voorwaarde is voldaan. Ook in dit geval moeten de verschillende procesbeschrijvingen worden vergeleken.

In dit verslag wordt een nieuwe grafische methode voor het weergeven van parallelle processen beschreven. Deze nieuwe methode is gebaseerd op de Nassy-Schneidermann-methode voor het weergeven van sequentiele processen, gecombineerd met de pijlen van de bollen en pijlen-methode.
In deze nieuwe methode wordt een proces weergegeven door een Nassy-Schneidermann-diagram. Tussen deze Nassy-Schneidermann-diagrammen lopen dan de pijlen die de interacties tussen de processen weergeven.
Bij deze methode is het wel duidelijk wanneer welke interacties precies plaatsvinden en welke processen op elkaar moeten wachten. Ook zijn voorwaardelijke structuren in een oogopslag te herkennen.
Deze nieuwe methode kan dan ook gebruikt worden, zodra de bollen en pijlen-methode te weinig informatie biedt en/of te onoverzichtelijk wordt.

INHOUDSOPGAVE

1. INLEIDING

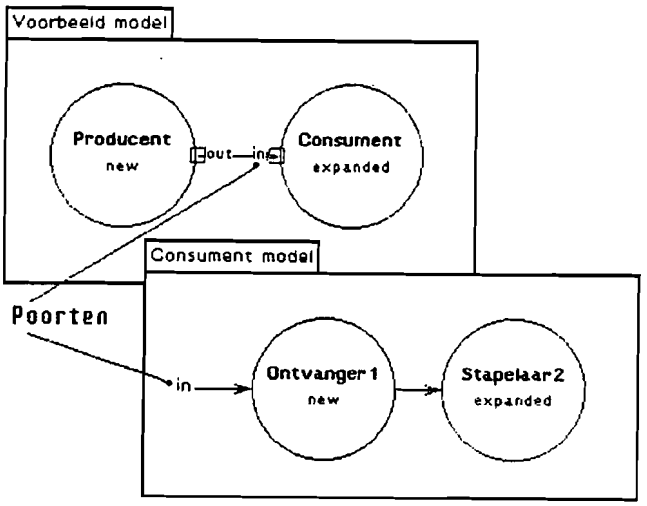
De proces-interactie-omgeving, zoals deze is geimplementeerd in de taal Smalltalk-80, wordt gebruikt voor de simulatie en besturing van parallelle processen.
Op het ogenblik worden de parallelle processen en hun interacties in deze programmeeromgeving grafisch weergegeven door bollen en pijlen [Wortmann, 1989]. Deze weergave geeft echter niet veel informatie over het verloop van het proces en wanneer welke interacties plaatsvinden. Het enige dat men aan zo'n bollen-pijlen-schema kan zien, is welke processen er zijn en of ze al dan niet met elkaar communiceren.
Wil men meer weten, bijvoorbeeld of bepaalde processen op elkaar moeten wachten, dan moet men de procesbeschrijving erbij halen, waarin exact staat beschreven, wat het proces doet en wanneer welke interacties plaatsvinden. Dit gaat goed zolang de processen nog tamelijk globaal worden beschreven.
Zodra de processen complexer worden, en er veel interacties tussen de processen plaatsvinden, gaat het overzicht over al de verschillende processen snel verloren. Ook indien de start van verschillende processen afhankelijk is van een bepaalde voorwaarde, dan kan men deze voorwaarde niet uit het bollen-pijlen-schema herkennen. Men zal weer de procesbeschrijvingen erbij moeten halen.

In dit verslag wordt een nieuwe grafische methode voor het weergeven van parallelle processen beschreven, die wel een overzicht over al de verschillende processen van een model geeft. Als een model nu zo gedetailleerd wordt, dat het overzicht verloren gaat, kan op de nieuwe methode worden overgegaan.

2. DE PROCES-INTERACTIE-OMGEVING

2.1. De grafische weergave van processen met bollen en pijlen

In de bestaande proces-interactie-omgeving, zoals die geimplementeerd is in Smalltalk-80, kunnen parallelle processen worden gesimuleerd en bestuurd. Deze parallelle processen en hun onderlinge interacties worden grafisch weergegeven met behulp van bollen en pijlen (zie figuur 2.1).
Een *bol* representeert hierin een proces dat onafhankelijk van de processen in de andere bollen kan verlopen, maar dat er gedeeltelijk ook van afhankelijk kan zijn. Deze onderlinge afhankelijkheid wordt weergegeven met behulp van de pijlen.
Een *pijl* representeert een synchrone interactie tussen twee processen. Synchroon betekent in dit geval dat het zendende proces moet wachten totdat het ontvangende proces de via de interactie verstuurde boodschap heeft ontvangen. Hierbij vindt dus een synchronisatie plaats van de twee processen. De richting van de pijl geeft duidelijk aan welk proces (gerepresenteerd door een bol) zendt en welk proces ontvangt.
Een pijl wordt aangesloten op de bollen met behulp van een *poort*. De poort heeft een voor de aangesloten bol unieke naam. Er kunnen ook meerdere pijlen op een poort worden aangesloten, maar iedere poort heeft hierbij maar een doorgangsrichting. Er kunnen alleen aankomende of alleen vertrekkende pijlen worden aangesloten, niet beiden tegelijk.
Tijdens een dergelijke interactie kan een object worden overgestuurd. Het proces genaamd 'Producent' uit figuur 2.1 zou bijvoorbeeld een door hem gegenereerd produkt over kunnen sturen naar het proces genaamd 'Consument', waarna het proces 'Consument' bepaalde acties kan verrichten met, of naar aanleiding van het verzonden produkt.
Ook kan een dergelijke interactie worden gebruikt om de verschillende processen te synchroniseren of om een bepaald proces door een ander proces te laten starten of stoppen.
Om het geheel van processen overzichtelijk en duidelijk gestructureerd te houden kan een proces worden geexpandeerd, zodra er parallellisme optreedt in zo'n proces. Een geexpandeerd proces bestaat dan weer uit verschillende processen met hun onderlinge interacties. Deze processen worden de kinderen van het geexpandeerde proces genoemd, terwijl het geexpandeerde proces de ouder van deze kinderen wordt genoemd. De kindprocessen en hun interacties worden ook weer gerepresenteerd door bollen en pijlen.
Het proces genaamd 'Consument' uit het voorbeeld is zo'n geexpandeerd proces, met een expansie bestaande uit een kindproces genaamd 'Ontvanger' en een kindproces genaamd 'Stapelaar' (zie ook fig. 2.1).
In de expansie worden de poorten van de bol die het ouderproces representeert, weergegeven door hun naam, zodat de pijlen die van en naar de bollen buiten de bol van het ouderproces leiden, aan deze poorten kunnen worden aangesloten.
De opdrachten die door een niet-geexpandeerd proces worden uitgevoerd, worden beschreven in de procesbeschrijving (zie fig. 2.2). Deze procesbeschrijving bestaat uit Smalltalk-80-code en de task-language van de proces-interactie-omgeving. Met behulp van deze task-language kan de programmeur onder meer de interacties tussen de processen specificeren.



figuur 2.1: Een voorbeeld van de grafische weergave van twee processen en hun interacties

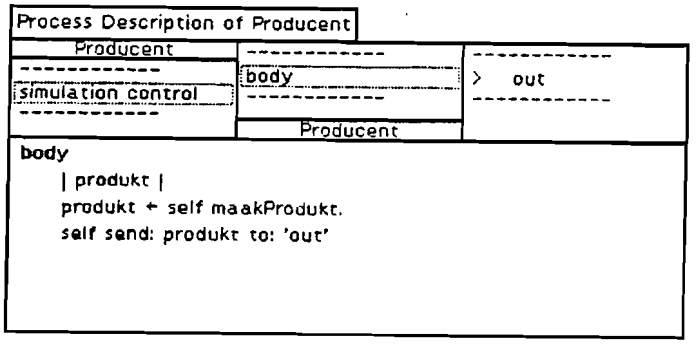


fig. 2.2: Een voorbeeld van een procesbeschrijving

Zie voor meer informatie over de programmeeromgeving Smalltalk-80 de boeken van Goldberg [1983, 1984 en 1989].
Zie voor een nadere uitwerking van de proces-interactie-benadering Rooda [1987] en Wortmann [1989 en 1990].

2.2. De beperkingen van de grafische weergave van processen met bollen en pijlen

De bestaande grafische weergave zoals beschreven in de vorige paragraaf heeft een aantal beperkingen. Het is om te beginnen niet mogelijk om, zonder de procesbeschrijving erbij te halen, te zien op welk punt in het proces de interacties plaatsvinden.
Nu is dit meestal geen groot probleem, omdat bij de processen in principe geen ingewikkelde procesbeschrijving hoort en men dus snel kan zien wat er in een proces precies gebeurt. Men kan in Smalltalk-80 namelijk ingewikkelde programmastructuren vervangen door één methode, die dan in de procesbeschrijving wordt aangeroepen [Goldberg, 1989]. Op deze wijze kan de procesbeschrijving zelf bijna altijd eenvoudig worden gehouden. Ook worden de processen geexpandeerd zodra er parallellisme plaatsvindt.

Bij voortgaande detaillering van het model belandt men echter al snel op een punt, waarbij het overzicht over al de verschillende processen verloren gaat. Het is dan niet meer duidelijk, wanneer de interacties precies plaatsvinden, welke processen wanneer op elkaar moeten wachten, welke opdrachten tegelijkertijd worden uitgevoerd, wanneer de verschillende processen worden gesynchroniseerd etc. Deze informatie is nog steeds beschikbaar in de procesbeschrijvingen, maar zij is alleen te verkrijgen door al de procesbeschrijvingen met elkaar te vergelijken. Dit kost veel tijd en vergissingen zijn snel gemaakt.

Een tweede beperking van de grafische weergave van processen en hun interacties met bollen en pijlen komt aan het licht zodra de uitvoer van meerdere parallelle processen afhankelijk is van dezelfde voorwaarde.

Een voorbeeld (zie figuur 2.3):

Een stapelaar moet produkten op een bepaalde stapel leggen.
Als het produkt blauw is moet het op de linker stapel worden gelegd, en als het rood is op de rechter. Het produkt wordt door een cylinder op de juiste stapel geschoven, waarna deze stapel door een andere cylinder iets omlaag wordt getrokken, zodat het volgende produkt erop kan worden geschoven. Tijdens het zakken van de stapel trekt de cylinder die het produkt erop heeft geschoven zich weer terug.
Voor iedere stapel moeten dus andere cylinders tegelijkertijd worden aangestuurd.

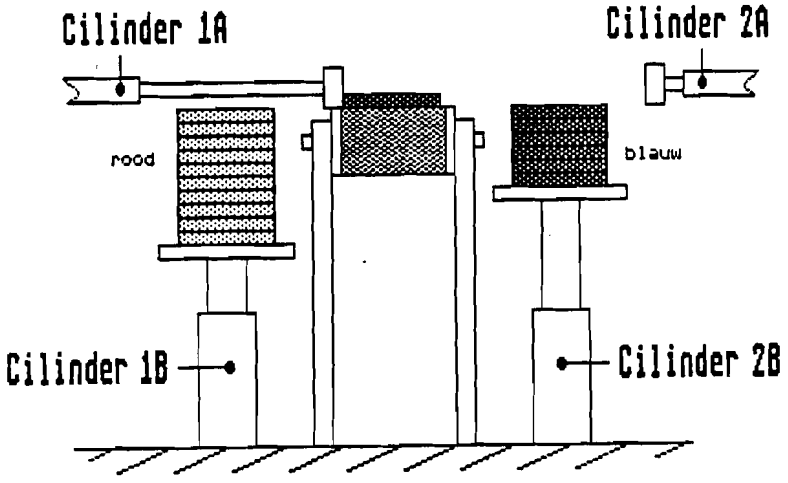


fig. 2.3: Een stapelaar

Met de bollen en pijlen-methode kunnen deze processen worden weergegeven door twee geexpandeerde bollen, voor iedere mogelijkheid van stapelen een.
Deze bollen bestaan dan ieder weer uit twee bollen, een voor het proces waarin het produkt op de juiste stapel wordt geschoven en voor het proces waarin de stapel omlaag wordt bewogen (zie figuur 2.4).
Dan moet er nog worden aangegeven welk van de stapel-processen mag worden uitgevoerd, afhankelijk van het binnengekomen produkt. Dit kan bijvoorbeeld worden geregeld door een extra proces, waarin wordt gekeken welke kleur het produkt heeft, waarna het naar het juiste stapelproces wordt gestuurd, maar ook kan het produkt naar beide processen worden gestuurd, waarna zij zelf controleren of het produkt de juiste kleur heeft.

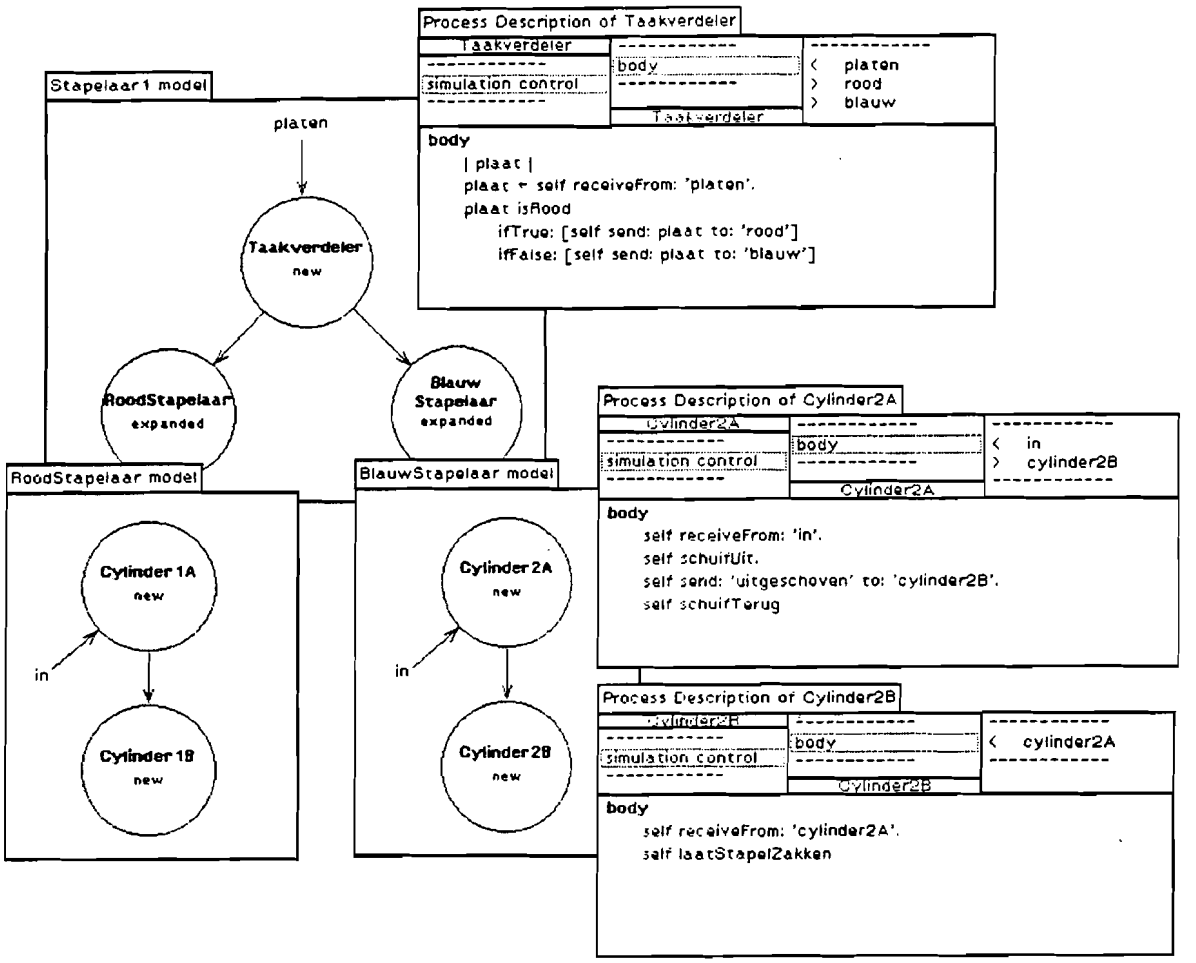


fig. 2.4: De besturing van de stapelaar zoals weergegeven met bollen en pijlen

Hoe het ook wordt opgelost, er zijn een heleboel bollen met korte procesbeschrijvingen voor nodig, en het geheel wordt erg complex. Het wordt moeilijk om snel te zien wat er nu eigenlijk precies gebeurd.
Men kan onder meer niet zien dat het verloop van het proces 'BlauwStapelaar' en van het proces 'RoodStapelaar' afhangen van verschillende voorwaarden. Dit ziet men pas als men de procesbeschrijving van de Taakverdeler erbij haalt.

Uit het voorgaande blijkt dat er een nieuwe grafische methode nodig is, die meer overzicht biedt dan de bollen en pijlen-methode. Deze methode zal dan de bollen en pijlen-methode kunnen vervangen zodra hiermee het overzicht over de processen verloren dreigt te gaan.

3. DE OPBOUW VAN EEN NIEUWE GRAFISCHE METHODE VOOR HET WEERGEVEN VAN PROCESSEN EN HUN INTERACTIES

Een nieuwe grafische methode die de bollen en pijlen-methode aan zal vullen, moet natuurlijk zoveel mogelijk de basis-principes en de mogelijkheden van de bollen en pijlen-methode behouden.
Dit betekent ondermeer dat interacties op dezelfde manier worden weergegeven en dat de procesbeschrijvingen op dezelfde manier en met dezelfde opdrachten moeten kunnen worden ingevuld. Ook moeten de parallelle processen kunnen worden geexpandeerd.
Daarnaast moet de methode zoveel mogelijk aansluiten op eventuele bestaande grafische programmeertechnieken, zoals bijvoorbeeld het flow-chart of het Nassy-Schneidermann-diagram [Jackson, 1983].

Uiteindelijk is gekozen voor een combinatie van de bollen en pijlen-methode met de Nassy-Schneidermann-methode.

3.1. Het Nassy-Schneidermann-diagram

Het Nassy-Schneidermann-diagram is oorspronkelijk ontwikkeld als hulpmiddel bij het gestructureerd ontwerpen van sequentiele computer-programma's.
De verschillende opdrachten, die tijdens zo'n programma worden uitgevoerd, worden weergegeven door blokjes (zie fig. 3.1).
Het programma heeft een duidelijk start- en eindpunt en het programma wordt van boven naar beneden doorlopen.
Daarbij kan een Nassy-Schneidermann-diagram nog enkele speciale blokken bevatten, die voorwaardelijke acties weergeven, namelijk het if..then-blok, het while..do-blok en het repeat..until-blok (zie fig. 3.2).
In principe kan met deze basis-elementen ieder programmmeerprobleem worden opgelost.

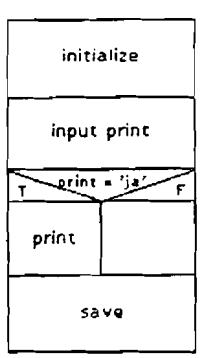


fig. 3.1: Een Nassy-Schneidermann-diagram

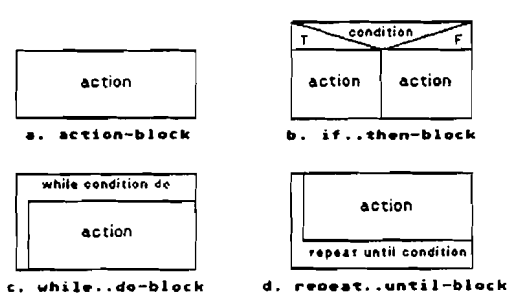


fig. 3.2: De basiselementen van een Nassy-Schneidermann-diagram

De voordelen van deze Nassy-Schneiderman-diagrammen ten opzichte van andere grafische methodes [Jackson, 1983], zijn in de context van dit verslag de volgende:
- De Nassy-Schneidermann diagrammen bevatten geen pijlen (dit in tegenstelling tot bijvoorbeeld flow-charts), die verwarring kunnen veroorzaken met de pijlen die interacties weergeven.
- De Nassy-Schneidermann-methode is al tamelijk bekend onder programmeurs. Zelfs als de methode niet bekend is, zijn de betekenissen van de verschillende grafische symbolen snel te leren. Het aantal nieuwe begrippen is ook zeer beperkt.
- De Nassy-Schneidermann-methode dwingt tot gestructureerd programmeren.
- De verschillende voorwaardelijke acties (if..then etc.) kunnen bij een Nassy-Schneidermann-diagram in een oogopslag worden herkend.

3.2. De combinatie van de Nassy-Schneidermann-methode met de bollen en pijlen-methode

De in de vorige paragraaf beschreven Nassy-Schneidermann-methode kan worden gecombineerd met de bollen en pijlen-methode. Deze nieuwe gecombineerde grafische techniek kan dan worden gebruikt zodra de in hoofdstuk 2 beschreven problemen zich voordoen, dus zodra de programmeur het overzicht over de verschillende processen dreigt te verliezen.
Op dat moment kan de programmeur bij het expanderen van een proces kiezen voor een weergave van de exapnsie de nieuwe grafische methode in plaats van de bollen en pijlen-methode. De kindprocessen in de expansie worden dan gerepresenteerd door kolommen in plaats van bollen. Iedere kolom wordt gevormd door een Nassy-Scheidermann-diagram (zie fig. 3.3).
De kolommen worden door de programmeer-omgeving automatisch oplopend genummerd van links naar rechts. Deze nummers vormen een hulpmiddel bij het weergeven van de expansie van een proces, zoals verderop in deze paragraaf zal worden verduidelijkt.
Ook geven deze nummers aan dat het proces, dat wordt gerepresenteerd door de kolom, steeds opnieuw zal worden doorlopen. Het nummer aan het einde van de kolom geeft hierbij de doorkoppeling naar het nummer aan het begin van de kolom weer.

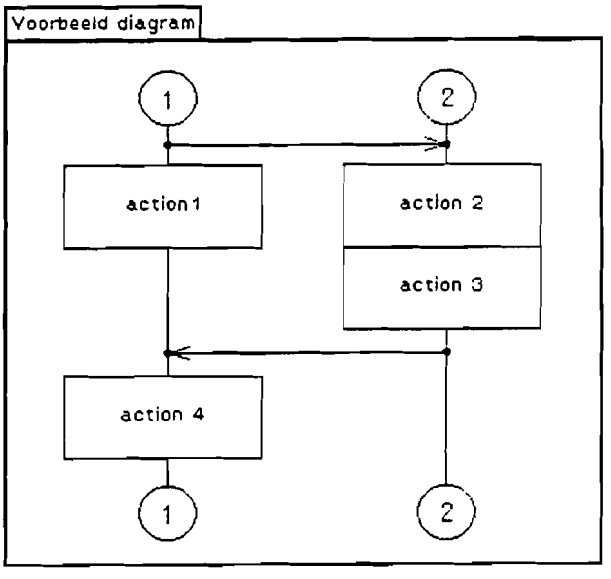


fig 3.3: Processen worden weergegeven door kolommen

De interacties tussen de processen worden net zoals bij de bollen en pijlen-methode weergegeven door pijlen. Deze interacties zijn dan synchroon. Eventuele asynchrone interacties, waarbij het zendende proces niet hoeft te wachten tot het ontvangende proces de boodschap heeft ontvangen, kunnen worden weergegeven door pijlen met een buffer ertussen (zie fig. 3.4).

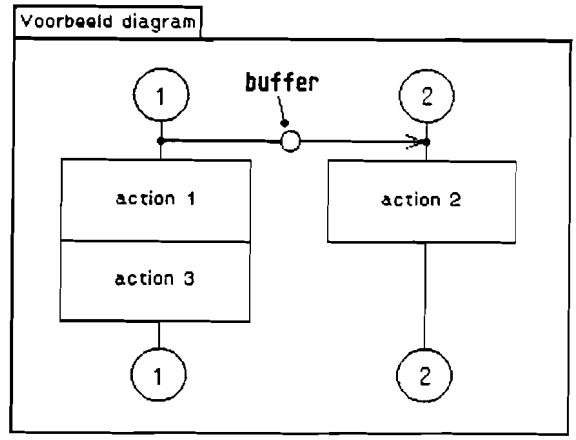


fig. 3.4: Een asynchrone interactie tussen twee processen

Bij deze nieuwe methode kunnen de pijlen op meerdere manieren aan de blokken worden aangesloten. Men zou ze namelijk aan kunnen sluiten op de boven- of benedenrand van het blok, op een van de zijranden, of op de flowlijn. De flowlijn wordt hierbij gedefinieerd als de lijn die de blokken van een kolom met elkaar verbindt (zie fig. 3.5).

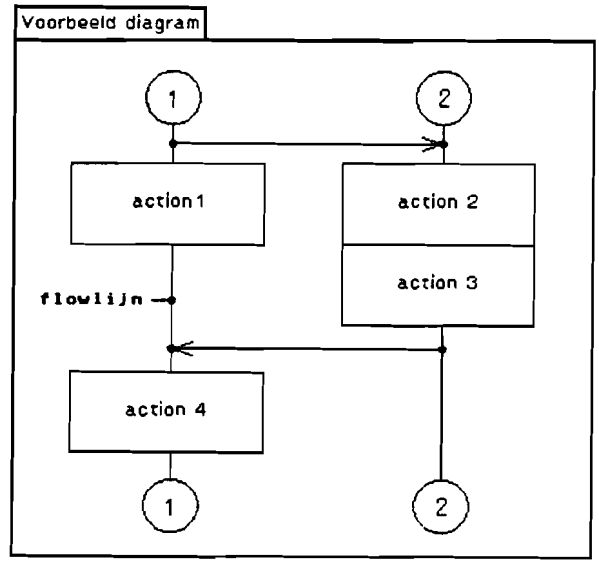


fig. 3.5: De flowlijn

Door de programmeer-omgeving wordt dan aangenomen, dat de pijlen die op de flowlijnen worden aangesloten, altijd interacties weergeven, die alleen maar dienen ter synchronisatie van de verschillende processen. De specificaties van deze interacties, kunnen dan door de programmeer-omgeving zelf worden gegenereerd. De programmeur hoeft deze interacties dus niet te specificeren.
Ook zal door de programmeer-omgeving worden verhinderd, dat er meer dan een pijl op hetzelfde punt van een flowlijn wordt aangesloten.
Dit alles zorgt ervoor, dat de synchronisaties tussen de verschillende processen in het kolommen-schema duidelijk te herkennen zijn. Ook de volgorde, waarin de verschillende synchronisaties plaatsvinden is meteen duidelijk. Zie ook figuur 3.6.

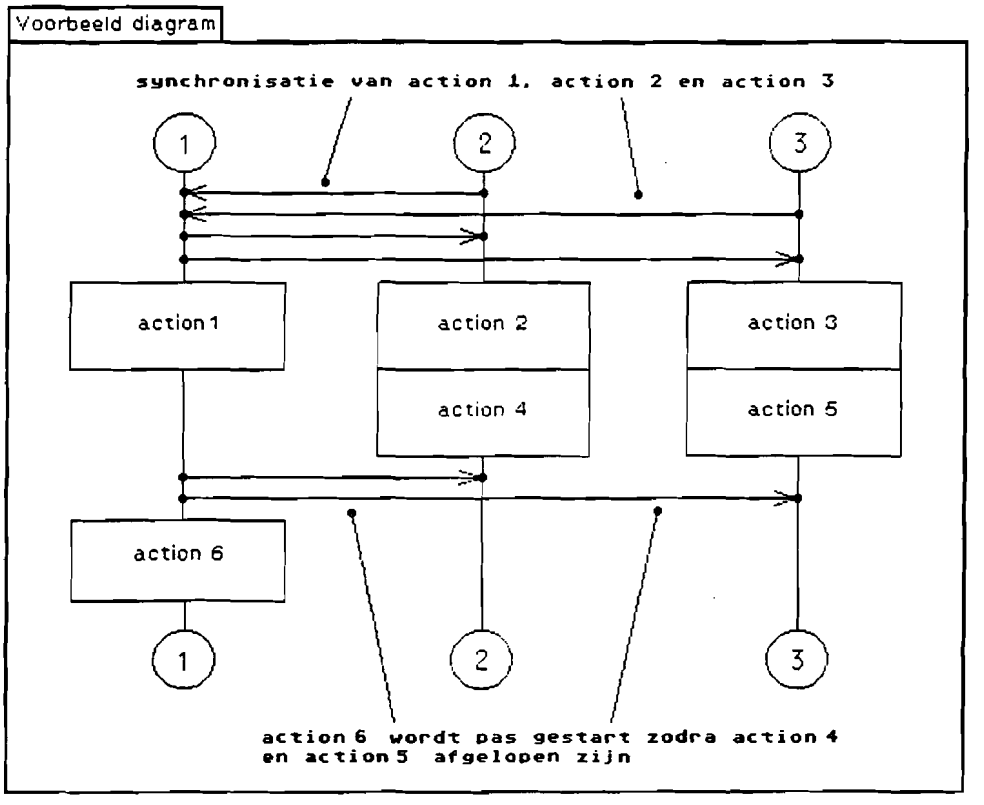


fig. 3.6: Synchronisatie van de verschillende processen

Als de programmeur echter een interactie tussen twee processen wil laten plaatsvinden, die niet alleen dient ter synchronisatie, bijvoorbeeld omdat hij een object wil meesturen, dan zal hij de pijl aan moeten sluiten op de desbetreffende blokken zelf, dus niet op de flowlijnen. Hierbij moet hij een poortnaam opgeven en de interactie met behulp van de task language in de actiebeschrijvingen van deze blokken specificeren. Zie ook figuur 3.7.

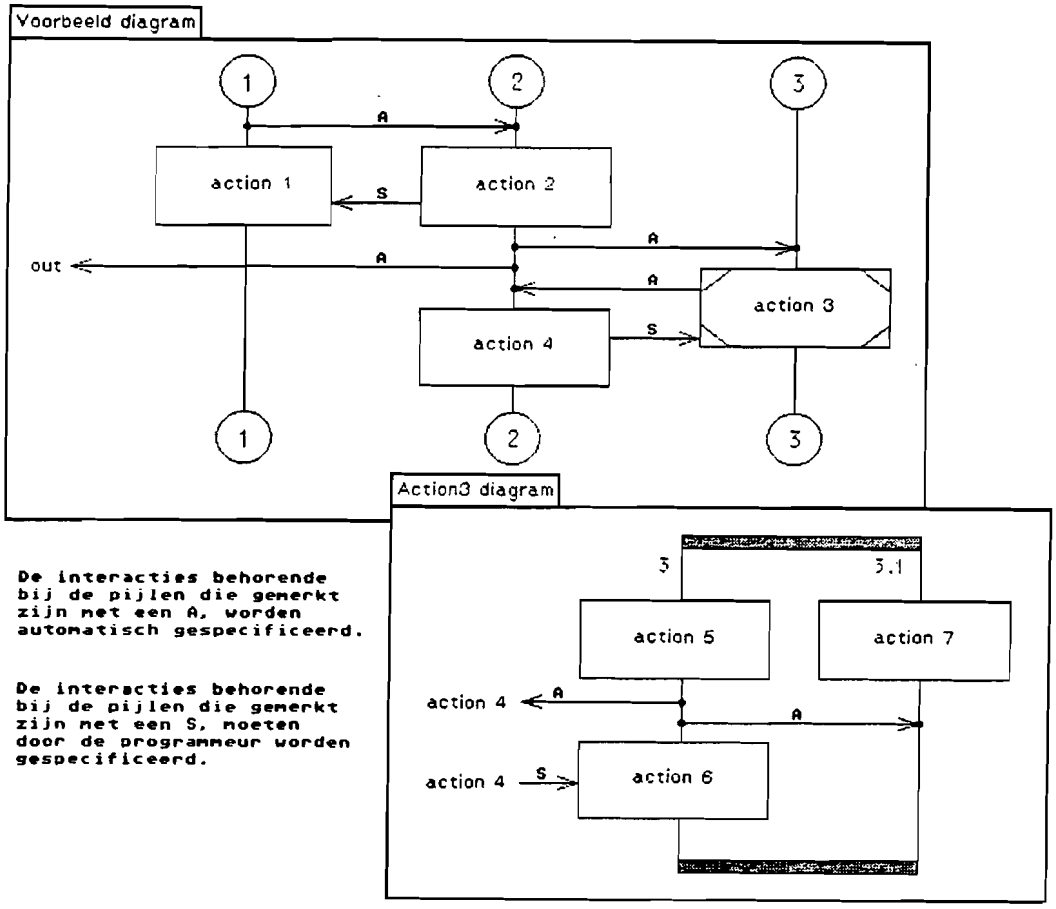


fig. 3.7: Bepaalde interacties worden automatisch gegenereerd

De poortnamen van de poorten op de flowlijn worden door de proces-interactie-omgeving gegenereerd. Deze poorten krijgen de naam 'col#port$', waarbij # wordt vervangen door het kolom-nummer en $ door het nummer van de poort. De poorten worden van boven naar beneden genummerd, beginnend met 1.
De poorten waarmee de pijlen op de bollen zijn aangesloten, worden op dezelfde wijze weergegeven als in de bollen en pijlen-methode. Dat wil zeggen dat de poorten die oorspronkelijk waren aangesloten op de geexpandeerde bol, worden weergegeven met hun naam, en dat de poorten die aangesloten zijn op de kolommen in de expansie naar keuze worden weergegeven met hun naam.

Een proces kan ook worden geexpandeerd, zodra er parallellisme in optreedt.
Vaak is het echter zo, dat dit parallellisme maar in enkele gedeelten van het proces voorkomt. De rest van het proces verloopt dan sequentieel.
Zo'n parallel gedeelte van het proces kan dan in de bijbehorende kolom worden weergegeven door een geexpandeerd blok, een blok met schuine lijntjes in de hoeken (zie fig. 3.8).
Bij zo'n geexpandeerd blok behoort een venster, dat kan worden geopend en waarin weer kolommen en pijlen kunnen worden geplaatst, die het op dat punt parallel verlopende gedeelte van het proces weergeven, dus ook het overeenkomstige deel van de kindprocessen. Deze kolommen worden in het vervolg sub-kolommen genoemd. Boven en onder de sub-kolommen wordt door de programmeer-omgeving automatisch een grijze balk geplaatst, die aangeeft dat de opdrachten in de sub-kolommen pas zullen worden uitgevoerd, zodra de het blok boven het bijbehorende geexpandeerde blok is afgelopen, en dat het blok onder het geexpandeerde blok pas zal beginnen, zodra de opdrachten en interacties in de subkolommen allemaal zijn uitgevoerd.
Zo kunnen er meerdere geexpandeerde blokken in een kolom worden geplaatst, die ieder een parallel verlopend gedeelte van het bijbehorende proces weergeven, en die in de bijbehorende vensters de overeenkomstige delen van de kindprocessen tonen.

Er kan echter ook een venster, het kolom-expansie-venster worden geopend, waarin de kindprocessen van het geexpandeerde proces geheel te zien zijn. Deze kindprocessen worden dan weergegeven door kolommen, die zijn samengesteld uit de sub-kolommen.
Hierbij worden alle sub-kolommen met hetzelfde nummer samengenomen. De programmeur bepaalt zelf het nummer van een sub-kolom, hij bepaalt dus ook welke sub-kolommen worden samengenomen. Dit geldt niet voor de eerste sub-kolom uit een blok-expansie-venster. Deze sub-kolom wordt in het kolom-expansie-venster namelijk in de plaats van het bijbehorende geexpandeerde blok gezet. Deze sub-kolom krijgt dan ook automatisch het nummer van de geexpandeerde kolom. De resterende sub-kolommen krijgen ook dit nummer. Daarachter wordt door de programmeer-omgeving een punt geplaatst en het nummer achter die punt wordt door de programmeur gekozen. Twee sub-kolommen met hetzelfde nummer in één blok-expansie-venster worden door de programmeer-omgeving niet toegestaan.
De programmeur bepaalt op deze manier dus ook welke instance-variabelen een blok uit een sub-kolom kan en mag gebruiken. Iedere kolom is hierbij een instance met zijn eigen variabelen [Goldberg, 1989].
In beide soorten vensters, dus in de blok-expansie-vensters en in het kolom-expansie-venster, kunnen kolommen, blokken en pijlen worden toegevoegd, verplaatst, gewijzigd en verwijderd. De veranderingen in het ene soort venster worden automatisch doorgevoerd in de andere vensters.
De grijze balken uit de blok-expansie-vensters worden ook op de overeenkomstige plaatsen in het kolom-expansie-venster gezet, met daarboven de naam van het bijbehorende geexpandeerde blok, zodat het duidelijk is waar de sub-kolommen beginnen en eindigen. De bij deze grijze balken behorende synchronisatie-interacties worden door de programmeer-omgeving automatisch gegenereerd.

Resumerend kan dus worden gesteld dat een proces, dat wordt weergegeven door een kolom, kan worden geexpandeerd in meerdere kindprocessen, die ieder weer worden weergegeven door een kolom in het kolom-expansie-venster. Het is echter vaak zo, dat maar in een gedeelte van een proces parallellisme optreedt. Dit gedeelte wordt dan weergegeven door een gexpandeerd blok. Als alleen maar zou worden aangegeven dat het gehele proces, dus de gehele kolom geexpandeerd is, dan zou het niet meteen duidelijk zijn dat een groot gedeelte van het proces sequentieel verloopt. Zie ook figuur 4.3 in hoofdstuk 4. Hier treedt in proces 3 parallellisme op, maar alleen in het klem-gedeelte van dat proces. De rest van het proces verloopt sequentieel. In eerste instantie is men waarschijnlijk niet geintereseerd in de details van dit klem-proces. Deze informatie blijft door de mogelijkheid van expansie dan ook verborgen tot men het blok-expansie-venster van het klemmen-blok opent. Een compleet overzicht van de kindprocessen van proces 3 verkrijgt men dan door het kolom-expansie-venster van kolom 3 te openen.

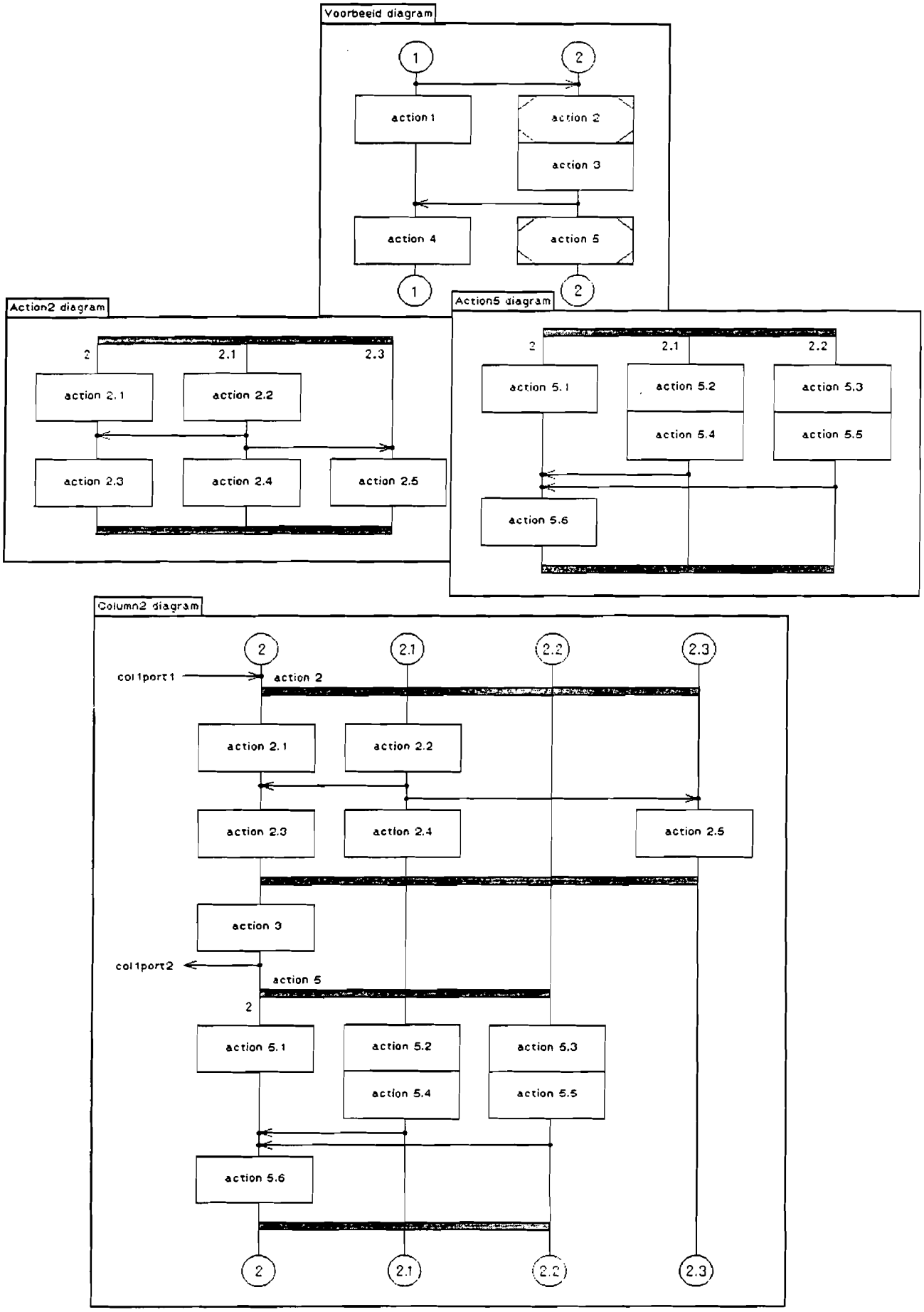


fig. 3.8: De weergave van een geexpandeerd blok en zijn expansie

Bij een niet-geexpandeerd blok hoort een actiebeschrijving, waarin de acties van het desbetreffende blok worden beschreven met behulp van Smalltalk-80-code en de task language uit de proces-interactie-omgeving, analoog aan de procesbeschrijving uit de bollen en pijlen-methode. Als deze actiebeschrijving maar een regel beslaat, kan zij ook worden opgenomen in de naam van het blok, zodat zij op het scherm te zien is.

Daarnaast is er nog de mogelijkheid om een kolom te initialiseren. Hierbij wordt een venster geopend, waarin men de verschillende instance-variabelen van de kolom kan initialiseren met behulp van Smalltalk-80-code. Deze initialisatie wordt eenmaal uitgevoerd, namelijk zodra het proces voor de eerste keer wordt opgestart. Bij elke volgende proces-loop, zal deze initialisatie worden overgeslagen. Deze initialisatie komt dus overeen met de 'initializeTasks'-method uit de bollen en pijlen-methode [Wortmann, 1990].
Ook kan men een venster openen waarin men aangeeft welke interacties en opdrachten moeten worden uitgevoerd voordat de verschillende processen starten. Door het systeem worden automatisch de nummers van alle kolommen, dus ook van de gexpandeerde kolommen onderin dit venster geplaatst. Boven deze nummers kunnen nu weer blokken en pijlen worden geplaatst. De opdrachten en interacties die door deze blokken en pijlen worden gerepresenteerd worden ook eenmaal uitgevoerd, net zoals de variabelen-initialisatie. Zij komen overeen met de 'initialActions'-method uit de bollen en pijlen-methode.
Zodra men een door een kolom weergegeven proces opstart, worden dus eerst de initialisatie-opdrachten uit het bijbehorende 'initializeTasks'-venster uitgevoerd, daarna de bijbehorende opdrachten en interacties uit het 'initialActions'-venster en dan pas de opdrachten en interacties uit de bij het proces behorende kolom. Nadat de kolom volledig is doorlopen, begint het proces weer bovenaan de kolom. De 'initialActions'- en de 'initializeTasks'-opdrachten worden pas weer uitgevoerd, zodra het proces na een reset opnieuw wordt opgestart.
Ter verduidelijking wordt in figuur 3.10 getoond hoe met deze nieuwe methode de stapelprocessen uit paragraaf 2.2 (figuur 2.3 en 2.4), de bijbehorende 'initializeColumns' en de bijbehorende 'ininitialActions' kunnen worden weergegeven.

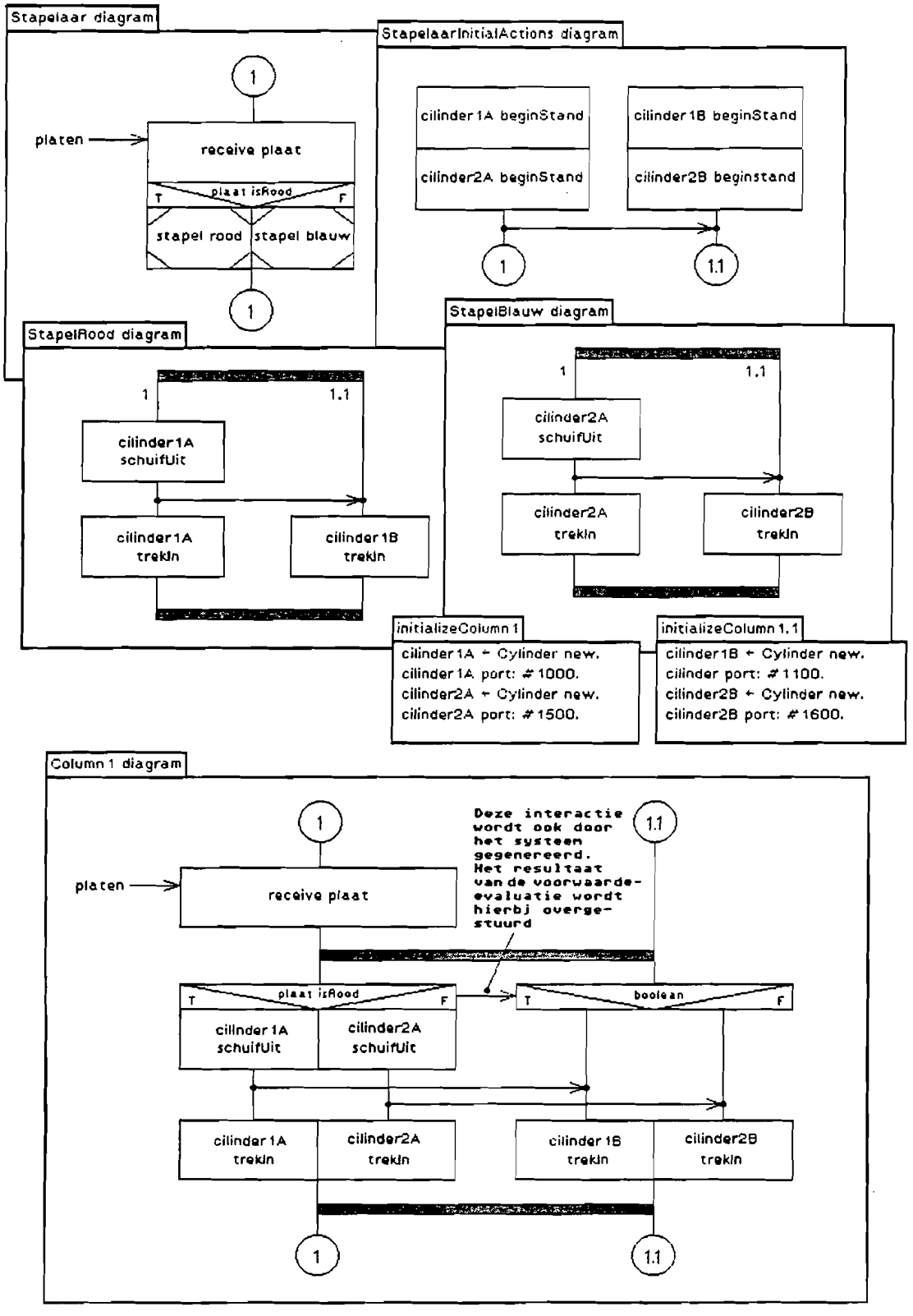


fig 3.9: De besturing van de stapelaar zoals weergegeven met de nieuwe grafische methode

Met deze nieuwe grafische methode worden de problemen zoals opgeworpen in hoofdstuk 2 inderdaad opgelost.

* Deze nieuwe methode biedt een totaal-overzicht over het gehele model. Het is duidelijk wanneer interacties plaatsvinden en tussen welke processen. Er wordt een duidelijk onderscheid gemaakt tussen synchroniserende interacties en interacties waarbij meer gebeurt, waarin bijvoorbeeld een object wordt overgebracht. Uit de kolommen-schema's wordt meteen duidelijk welke blokken tegelijkertijd worden uitgevoerd en welke zeker niet. Ook worden overbodige synchronisatie-interacties veel eenvoudiger opgespoord.
* Meerdere parallelle processen wier start afhankelijk is van dezelfde voorwaarde kunnen worden weergegeven door een geexpandeerd blok in een voorwaardelijk blok. Op deze manier is in een oogopslag te zien welke processen van een bepaalde voorwaarde afhankelijk zijn.

Tenslotte wordt via deze nieuwe grafische methode, tijdens de uitvoering van de met de kolommen beschreven processen, voortdurend weergegeven waar het proces mee bezig is. Een actieblok krijgt namelijk een grijze arcering zodra het proces de opdrachten uitvoert die in de actiebeschrijving van dat blok zijn beschreven. Als het proces op een bepaalde interactie staat te wachten, dan zal om het aansluit-punt van de bijbehorende pijl een vierkantje worden geplaatst.

In het volgende hoofdstuk wordt deze nieuwe grafische methode en zijn voordelen ten opzichte van de bollen en pijlen-methode verduidelijkt aan de hand van een paar voorbeelden, waarbij de oorspronkelijke processen-modellering met behulp van de bollen en pijlen-methode wordt getoond naast de modellering met de nieuwe grafische methode.

4. EEN VOORBEELD VAN HET GEBRUIK VAN DE NIEUWE GRAFISCHE METHODE.

Voor dit voorbeeld is gekeken naar de besturing van een stomplasautomaat voor het aan elkaar lassen van rubber binnenbanden, zoals die wordt gebruikt bij de firma Vredestein in Doetinchem.
Door een invoer-proces wordt een hol stuk rubber aangevoerd. Dit stuk rubber wordt door de stomplasautomaat vastgepakt en de uiteinden van het rubber worden op de juiste maat gesneden. Daarna worden deze uiteinden onder hoge druk tegen elkaar geklemd. Onder deze hoge druk vloeien de uiteinden in elkaar over. Het holle stuk rubber is een binnenband geworden, die daarna uit de machine wordt genomen.
Bij de besturing van deze stomplasautomaat vinden een aantal kortdurende parallelle acties plaats. Deze acties kunnen nu worden beschreven met de nieuwe grafische methode.

In figuur 4.1 wordt het geheel van de processen getoond, die met de stomplasautomaat in verband staan. Zij kunnen nog gewoon worden gerepresenteerd door de bollen en pijlen uit de proces-interactie-omgeving.
De geexpandeerde bol genaamd 'SLA' representeert het eigenlijke stomplasproces. In figuur 4.2 wordt de expansie van dit proces weergegeven met behulp van de bollen en pijlen-methode. Ook wordt de procesbeschrijving van de verschillende bollen in deze expansie getoond.
In figuur 4.3 wordt de expansie van hetzelfde proces getoond, maar nu beschreven met de nieuwe grafische methode.
Bij deze beschrijving werd als beperking aangehouden, dat ieder machine-onderdeel maar in een kolom mocht voorkomen. Eerst werd hierbij voor ieder machine-onderdeel een aparte kolom gemaakt. Daarna werden de kolommen van machine-onderdelen die toch nooit tegelijkertijd bewegen, zoals de klemmen en de bandafpellers, samengevoegd om een compacter en overzichtelijker geheel te verkrijgen.
Het proces dat wordt gerepresenteerd door een kolom verloopt sequentieel, dus de niet-geexpandeerde blokken die in een kolom boven elkaar staan zouden kunnen worden samengenomen. Zij zouden dan samen een blok vormen, met een actiebeschrijving die de actiebeschrijvingen van de verschillende losse blokken combineert.
In dit voorbeeld werd echter iedere afzonderlijke opdracht (een Smalltalk-80 message) in een apart blok geplaatst, om het verloop van het proces duidelijk te kunnen maken zonder van ieder blok de actiebeschrijvingen te hoeven tonen.

N.B: Deze processen geven niet exact de besturing van de stomplasautomaat weer. Enkele kleinere onderdelen zijn uit de processen weggelaten wegens plaatsgebrek. Dit voorbeeld wil echter ook alleen maar een indruk geven van het verschil tussen de grafische weergave van processen met bollen en pijlen en de weergave van diezelfde processen met kolommen en pijlen.

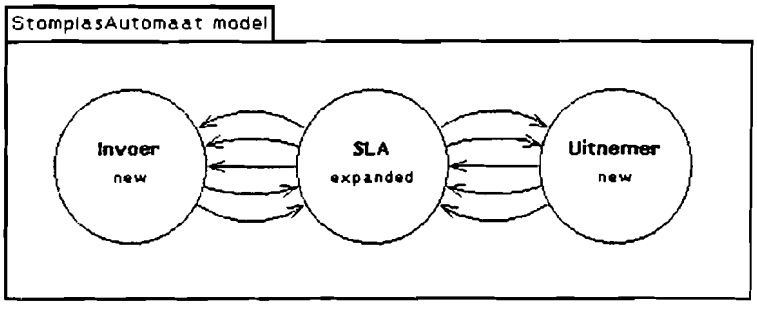


fig. 4.1: De processen die in verband staan met de stomplasautomaat

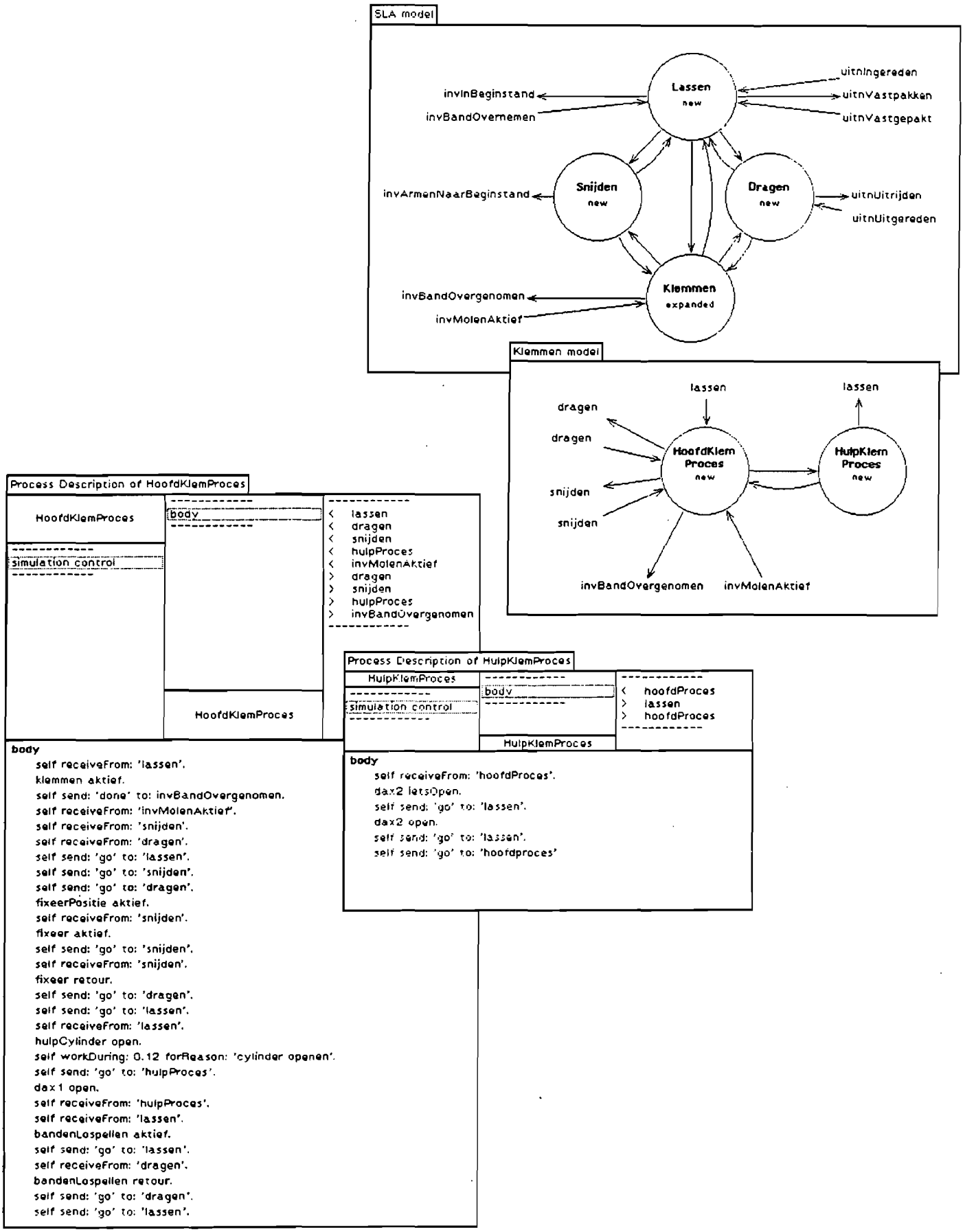


fig. 4.2: De expansie van de SLA-bol, zoals beschreven met bollen en pijlen

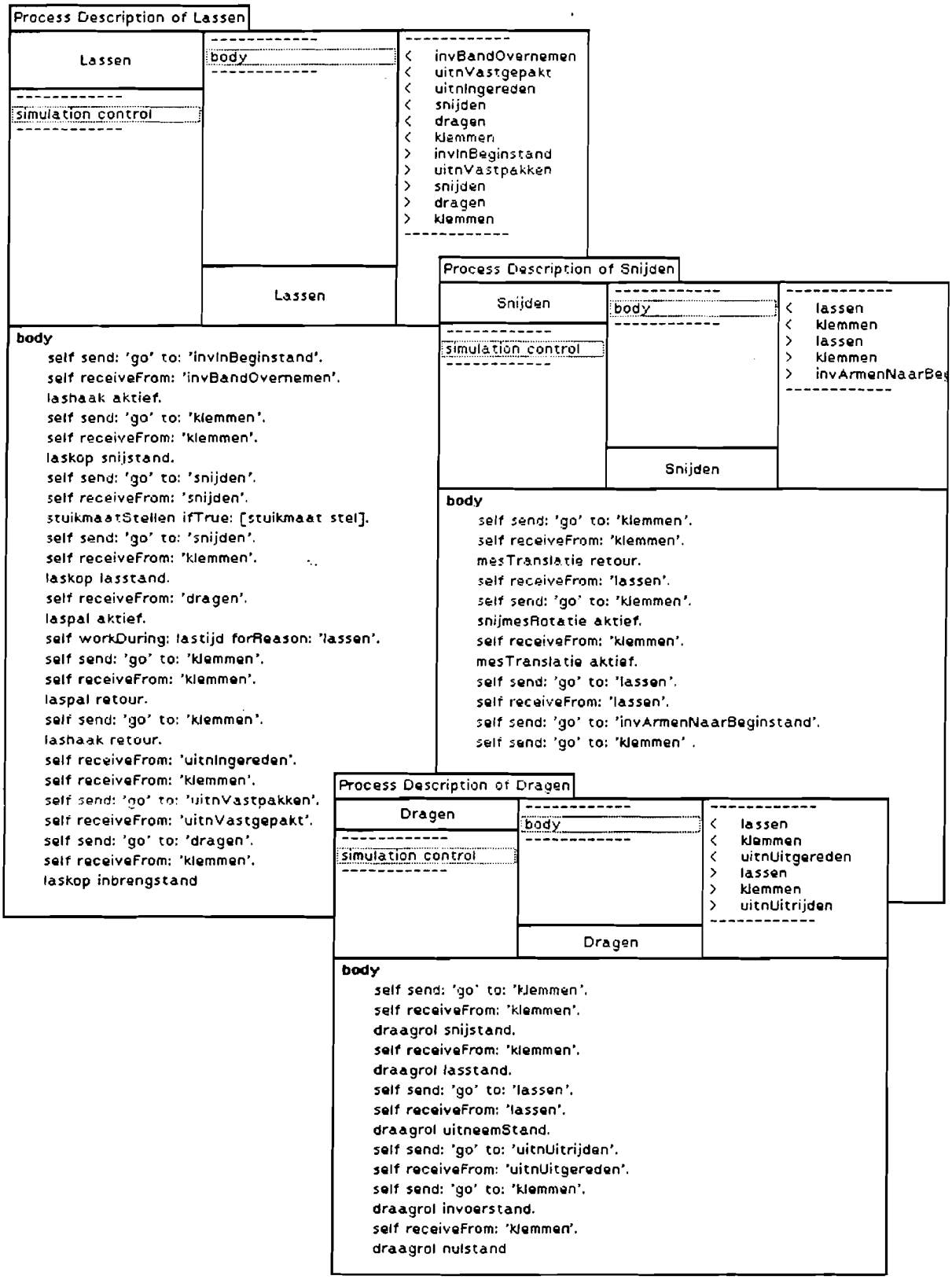


fig. 4.2 (vervolg): De expansie van de SLA-bol, zoals beschreven met bollen en pijlen

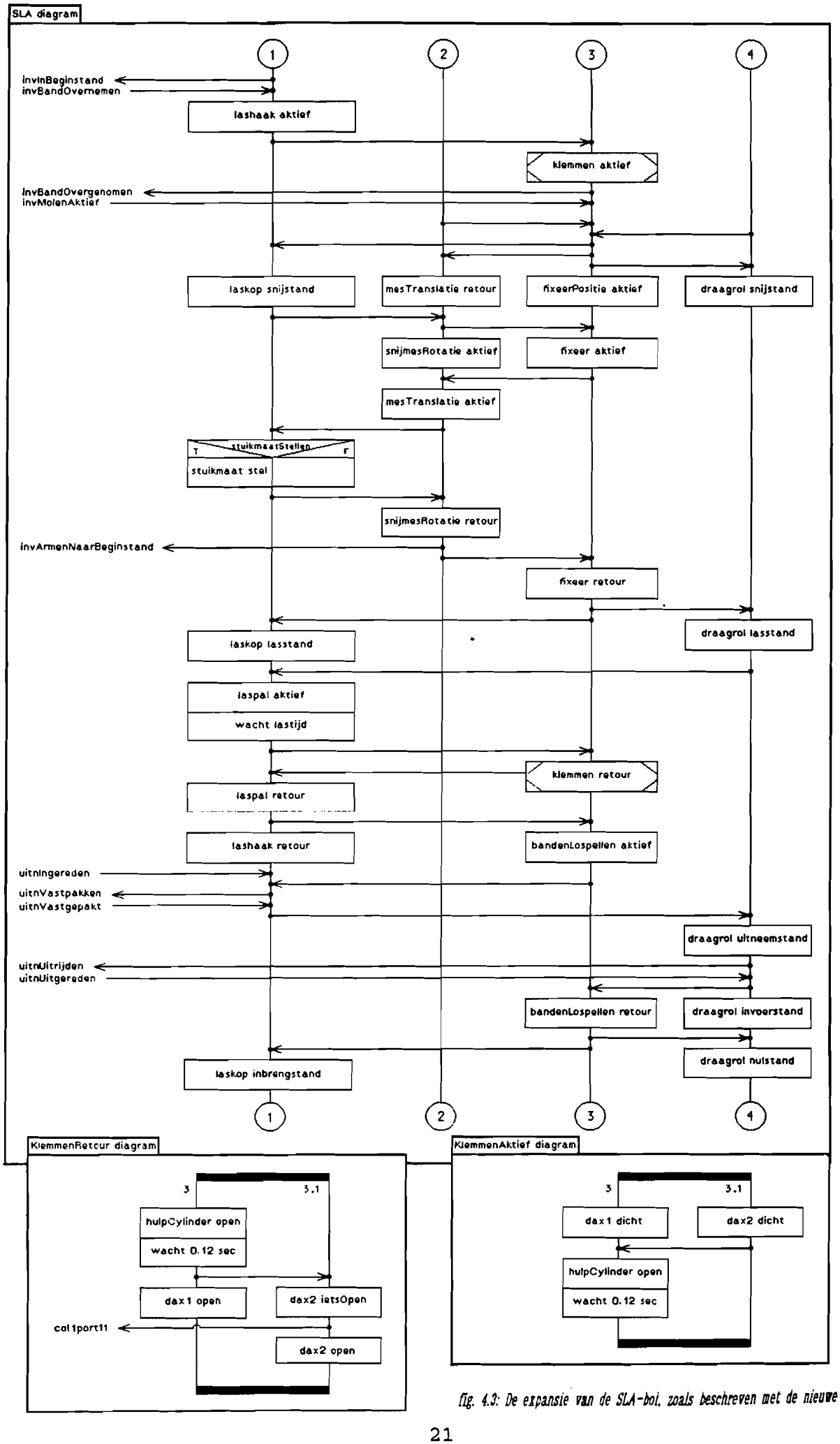


fig. 4.3: De expansie van de SLA-boi, zoals beschreven met de nieuwe grafische methode

5. DE VERTALING VAN DE MET DE NIEUWE GRAFISCHE METHODE BESCHREVEN PROCESSEN NAAR SMALLTALK-80-CODE

De met de kolommen en pijlen beschreven processen moeten nu nog worden vertaald naar Smalltalk-80-code. Zij worden op dezelfde manier vertaald als de processen beschreven met bollen en pijlen.
Dat wil zeggen dat voor iedere kolom een subclass van Bubble wordt gegenereerd. Deze subclass krijgt hierbij de naam van de bol die oorspronkelijk was geexpandeerd. Aan deze naam wordt een letter toegevoegd die overeenkomt met het nummer van de kolom. 1 wordt hierbij A, 2 wordt B, 2.1 wordt BA, 2.2 wordt BB etc. De kolom met nummer 1.1 uit figuur 3.9 (hoofdstuk 3) wordt dan ook vertaald naar de class genaamd 'StapelaarAA'.
De methods waarin de actiebeschrijvingen van de blokken zijn gedefinieerd (zie het einde van paragraaf 3.2), worden in deze class geplaatst.
In deze vertaalde classes worden nog drie instance-methods gegenereerd. In de instance-method genaamd 'body' worden na elkaar de actiebeschrijvingen van de blokken opgeroepen. Tussen deze actie-beschrijvingen worden in de 'body'-method de specificaties van de synchronisatie-interacties geplaatst, die behoren bij de pijlen aangesloten op de flowlijnen van de kolom en die behoren bij de grijze balken in de eventuele expansie-vensters van de kolom.
De actie-beschrijvingen van de blokken en de specificaties behorende bij de pijlen uit het 'initialActions'-venster worden ingevuld in de instance-methode genaamd 'initialActions'.
De opdrachten uit het 'initializeColumn'-venster worden ingevuld in de instance-method genaamd 'initializeTasks'.
Zodra een proces worden opgestart wordt de bijbehorende 'initializeTasks'-methode uitgevoerd, daarna de 'initialActions'-methode en daarna continu de 'body'-methode.

Ter illustratie worden in figuur 5.1 de 'initializeTasks'-, de 'initialActions'- en de 'body'-method getoond, die door de programmeer-omgeving zijn gegenereerd uit het stapelaar-diagram (zie fig. 3.9, hoofdstuk 3). Ook worden in deze figuur de verschillende methods getoond, waarnaar de actie-beschrijvingen van de blokken zijn vertaald en die in de 'body'- en 'initializeTasks'-method worden opgeroepen.

```
Bubble subclass: #StapelaarA
    instanceVariableNames: 'cilinder1A cilinder2A plaat'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Simulation-Stapelaar'

---------- StapelaarA ) action descriptions ----------

block1
    plaat <- self receiveFrom: 'platen'

falseBlock1
    cilinder2A schuifUit

falseBlock2
    cilinder2A trekIn

initialBlock1
    cilinder1A beginStand

initialBlock2
    cilinder2A beginStand

trueBlock 1
    cilinder1A schuifUit

trueBlock 2
    cilinder1A trekIn

---------- StapelaarA ) simulation control ----------

body
    self block1.
    self send: 'go' to: 'coll:p1Port1'.
    self send: (plaat isKnod) to: 'coll:p1BooleanPort1'.
    plaat isKnoop
        ifTrue:
            [self trueBlock1.
            self send: 'go' to: 'coll:p1Port2'.
            self trueBlock2]
        ifFalse:
            [self falseBlock1.
            self send: 'go' to: 'coll:p1Port3'.
            self falseBlock2].
    self send: 'go' to: 'coll:p1Port4

initialActions
    self initialBlock1.
    self initialBlock2.
    self send: 'go' to: 'coll:p1InitialPort1'

initializeTasks
    cilinder1A <- Cylinder new.
    cilinder1A port: #1000.
    cilinder2A <- Cylinder new.
    cilinder2A port: #1500
```

```
Bubble subclass #StapelaarAA
    instanceVariableNames: 'cilinder1B cilinder2B'.
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Simulation-Stapelaar'

---------- StapelaarAA ) action descriptions----------

falseBlock1
    cilinder2B trekIn

initialBlock1
    cilinder1B beginStand

initialBlock2
    cilinder2B beginStand

trueBlock1
    cilinder1B trekIn

----------StapelaarAA)simulationcontrol----------

body
    self receiveFrom: 'coll:Port1'.
    (self receiveFrom: 'coll:BooleanPort1')
        ifTrue:
            [self receiveFrom: 'coll:Port2'.
            self trueBlock1]
        ifFalse:
            [self receiveFrom: 'coll:Port3'.
            self falseBlock1].
    self receiveFrom: 'coll:Port4'

initialActions
    self initialBlock1.
    self initialBlock2.
    self receiveFrom: 'coll:InitialPort1'

initializeTasks
    cilinder1B <- Cylinder new.
    cilinder1B port: #1100.
    cilinder2B <- Cylinder new.
    cilinder2B port: #1600
```

fig. 5.1: De Smalltalk-80-code behorende bij het stapelaar-diagram

6. CONCLUSIE

De in dit verslag beschreven grafische methode is inderdaad in staat om de in hoofdstuk 2 beschreven beperkingen van de bollen en pijlen-methode op te heffen. Deze nieuwe methode geeft dan ook veel meer informatie over de processen dan de bollen en pijlen-methode. Bij deze nieuwe methode is het onder meer in een oogopslag duidelijk hoe elk proces precies verloopt en wanneer welke interacties plaatsvinden. Men ziet ook meteen welke voorwaardelijke structuren er zijn en welke processen op elkaar moeten wachten.

Nu is de vraag of dit betekent dat de bollen en pijlen-methode geheel vervangen moet worden door deze nieuwe methode. In principe zou dit wel kunnen, maar waarschijnlijk biedt deze nieuwe methode juist teveel informatie als de processen maar globaal beschreven worden. Vaak heeft men genoeg aan de informatie die de bollen en pijlen-methode biedt, dus welke processen er zijn en in welke mate ze samenghangen. Als men meer over een proces wil weten dan haalt men de proces-beschrijving van dit proces erbij en men kan dan precies zien wat er in dat ene proces gebeurt. Over de andere processen weet men dan niets en men hoeft dit ook vaak niet te weten.

Het voordeel van de combinatie van deze nieuwe methode met de bollen en pijlen-methode is dan ook, dat de programmeur zelf kan kiezen aan welke methode hij de voorkeur geeft.
Zodra de bollen en pijlen-methode te onoverzichtelijk wordt of te weinig informatie biedt kan hij dan overgaan op de nieuwe methode.

LITERATUURLIJST

Goldberg A., Robson D.,
Smalltalk-80, the Language.
Addison-Wesley Publishing Company, Menlo Park, 1989.

Goldberg A., Robson D.,
Smalltalk-80, the Language and its Implementation.
Addison-Wesley Publishing Company, Menlo Park, 1983.

Goldberg A., Robson D.,
Smalltalk-80, the Interactive Programming Environment.
Addison-Wesley Publishing Company, Menlo Park, 1984.

Jackson, M.A.
System Developement.
Prentice-Hall International, London, 1983

Rooda, J.E., Boot, W.C.,
Procescomputers 2.
Dictaat nr: 4635,
Technische Universiteit Eindhoven, 1987.

Wortmann A.M., Rooda, J.E., Boot W.C.,
Basisbegrippen van de Proces-Interactie-Benadering,
Rapportnr: WPA 0657,
Technische Universiteit Eindhoven, 1989

Wortmann A.M., Rooda, J.E.,
The Process-Interaction-Environment User Manual.
Rapportnr: WPA 0841,
Technische Universiteit Eindhoven, 1990.