

FLOWer scoort uitstekend op workflow patterns

Pallas Athena
Process Management

Inhoudsopgave

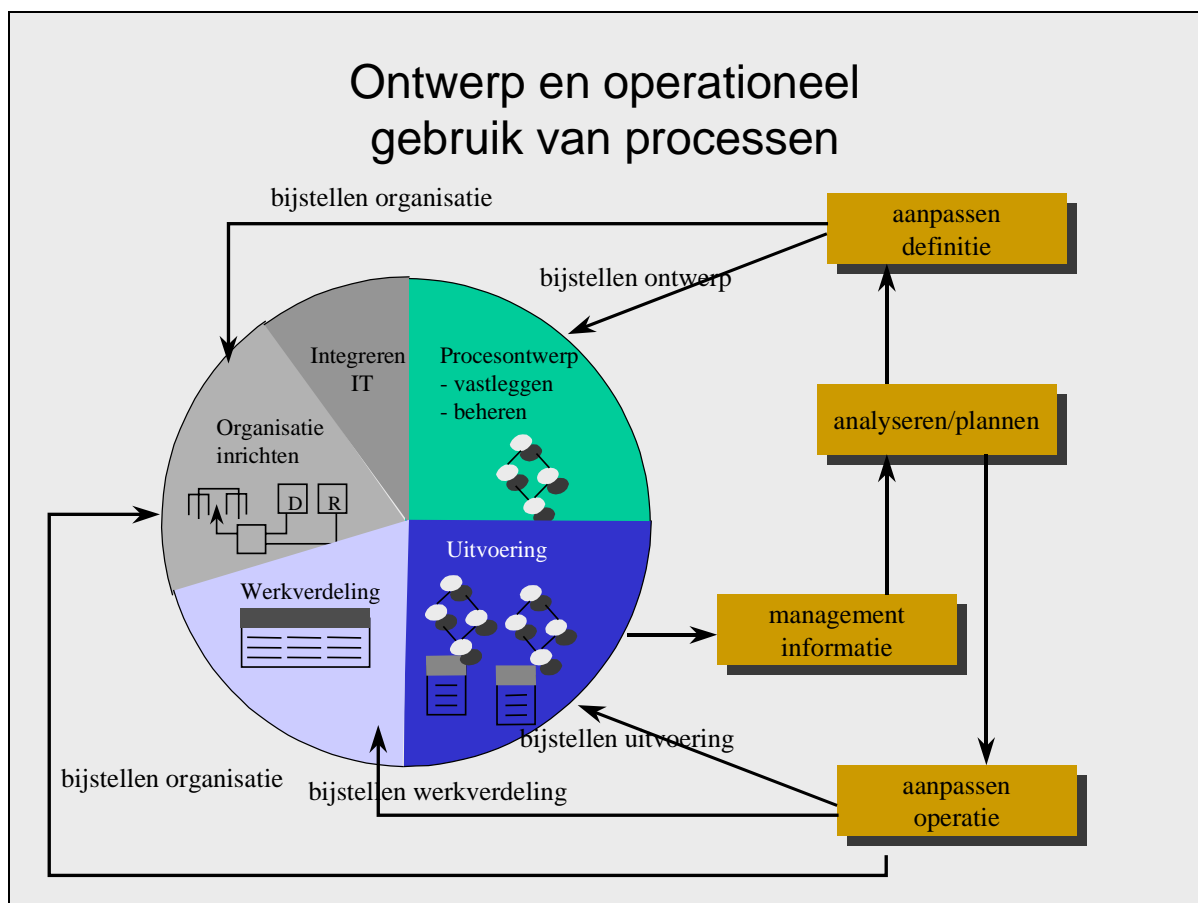
1	Inleiding	1
2	Het kader van het onderzoek.....	1
3	<i>FLOWer</i> model.....	2
4	De patronen	3
4.1	Pattern 1. Sequence.....	3
4.2	Pattern 2. Parallel split.....	3
4.3	Pattern 3. Synchronisation.....	3
4.4	Pattern 4. Exclusive Choice.....	3
4.5	Pattern 5. Simple merge.....	4
4.6	Pattern 6. Multi-Choice	4
4.7	Pattern 7. Synchronizing merge.....	4
4.8	Pattern 8. Multi-merge.....	5
4.9	Pattern 9. Discriminator.....	6
4.10	Pattern 10. Arbitrary cycles	7
4.11	Pattern 11. Implicit termination.....	7
4.12	Pattern 12. Multiple instances without synchronization.....	9
4.13	Pattern 13. Multiple instances with a priori design time knowledge.....	9
4.14	Pattern 14. Multiple instances with a priori runtime knowledge.....	9
4.15	Pattern 15. Multiple instances without a priori runtime knowledge.....	9
4.16	Pattern 16. Deferred choice	9
4.17	Pattern 17. Interleaved parallel routing	10
4.18	Pattern 18. Milestone.....	10
4.19	Pattern 19. Cancel activity.....	10
4.20	Pattern 20. Cancel case.....	10
5	Conclusies	11
6	Literatuur.....	12

1 Inleiding

Op de markt is een groot aantal workflow management systemen beschikbaar. Ondanks pogingen te komen tot uniformiteit in modellering en uitwisselbaarheid, met name door de Workflow Management Coalition (WfMC), zijn de verschillen tussen deze systemen groot. Op het eerste gezicht kunnen alle systemen ongeveer hetzelfde. De voor de hand liggende modelleerconstructies zoals parallelle paden en sequenties van activiteiten kunnen alle systemen naar behoren invullen en vormen dus geen geschikte selectiecriteria. Voor een goede vergelijking dient men in detail te kijken naar de werkelijke toepasbaarheid van de systemen. In Workflow Patterns [WPA] wordt op wetenschappelijke basis een onderzoek gedaan naar de mate waarin een vijftiental bestaande workflow management systemen veel voorkomende patronen ondersteunt. Het gaat daarbij dus om de *expressiekracht* van de *procesmodelleeromgeving*. In dit artikel gaan we dieper in op de resultaten van FLOWer in het onderzoek. FLOWer is het workflow management en case handling systeem van Pallas Athena. Dit artikel is bedoeld voor iedereen die te maken heeft met de selectie van workflow management systemen of meer algemeen met de toepassing van procesbesturing. We raden de lezer aan [WPA] bij de hand te houden als achtergrondinformatie.

2 Het kader van het onderzoek

In het onderzoek naar workflow patronen [WPA] ligt de focus op de procesmodelleeromgeving. Die aanpak leidt tot de identificatie van een minimaal aantal *noodzakelijke* voorwaarden voor de selectie van een workflow management systeem. Door de focus van het onderzoek blijven allerlei eveneens belangrijke (en noodzakelijke) voorwaarden buiten beschouwing. We bespreken eerst in vogelvlucht die andere belangrijke aspecten van procesbesturing, zoals weergegeven in figuur 1. Vervolgens bespreken we het onderzoek naar workflow patronen.



figuur 1. De verschillende aspecten van procesbesturing

De verschillende aspecten van procesbesturing kunnen het beste gelezen worden tegen de klok in, te beginnen bij de taartpunt rechtsboven: *procesontwerp*. De workflow patterns uit [WPA] zijn daar te plaatsen. Overigens is die taartpunt niet volledig beschreven met de patronen. Aspecten die niet aan de orde komen zijn bijvoorbeeld:

- Het autorisatiemodel om bevoegdheden aan te geven;

- De ondersteuning van een product- of datagedreven aanpak, waarbij de status en voortgang van een zaak bepaald kunnen worden uit de aanwezige informatie.

In de andere taartpunten komen bijvoorbeeld de volgende zaken aan de orde:

- Ondersteuning van de operationele flexibiliteit, waarbij de gebruiker, indien nodig, voldoende vrijheid van handelen wordt geboden. Vanzelfsprekend dient dit ook voor een deel in het ontwerp meegenomen te worden;
- De wijze van presentatie aan de eindgebruiker;
- De mate waarin uitzonderingen ondersteund kunnen worden;
- Het gebruikte distributiemodel om zaken te verdelen over de medewerkers;
- De wijze waarop autorisatie en distributie gescheiden kunnen worden, bijvoorbeeld om in organisaties de werkverdeling en de wijze van afhandeling ook lokaal te kunnen regelen;
- De wijze waarop de organisatie kan worden ingericht;
- De schaalbaarheid van het systeem;
- De integreerbaarheid van het systeem met andere systemen en applicaties.

En tenslotte zijn er nog andere aspecten van belang zoals het leidende paradigma van het systeem. Daarbij gaat het bijvoorbeeld om de vraag of het systeem routing- of activiteitgeoriënteerd is, of het systeem de zaak centraal stelt (*case handling*) of de losse activiteiten. Die aspecten zijn cruciaal voor een succesvolle toepassing en dienen in een selectietraject meegenomen worden.

Voor een gedetailleerde beschrijving van alle genoemde aspecten verwijzen we naar [FPP].

We beperken ons in dit artikel tot de *expressiekracht* van de *procesontwerpomgeving* aan de hand van de in [WPA] genoemde patronen. In het volgende hoofdstuk gaan we kort in op het onderzoek en de betekenis van de resultaten. In hoofdstuk 4 bespreken we alle patronen. In hoofdstuk 5 geven we de conclusies.

Het onderzoek [WPA] is reeds in 2000 uitgevoerd. In de laatste versie (februari 2002, draft version) is voor het eerst ook FLOWer bekeken. FLOWer is het eerste *case handling* systeem. In een *case handling* systeem staat de zaak centraal en niet een losse activiteit, zoals in de traditionele workflow management systemen. Voor een uitgebreide beschrijving van FLOWer en de achterliggende filosofie verwijzen we naar [FPP], te vinden op de Pallas Athena site www.pallas-athena.com.

Een uittreksel van de resultaten van FLOWer, rechtstreeks overgenomen uit de voorlopige versie van [WPA] treft u aan bij de bespreking van de patronen.

Het is van belang de wetenschappelijke achtergrond van het onderzoek in het oog te houden. Een pakket ondersteunt een patroon *direct* indien het patroon in de modelleeromgeving *getekend* kan worden. Indien er bijvoorbeeld sprake is van het gebruik van extra condities, het toevoegen van verschillende modelleerniveaus of het gebruik van contextspecifieke informatie, is de beoordeling per definitie nooit positief (+). We tekenen daarbij aan dat in het rapport weliswaar geen sluitende definitie van “direct” wordt gegeven, maar dat het zeker leidt tot (wetenschappelijk) goed vergelijkbare producten. Wel kan de betekenis van een negatieve beoordeling (aangegeven met een -) nogal verschillen. In het ene geval betekent het dat er geen directe ondersteuning is, maar er wel een isomorfe of gelijkwaardige constructie in het pakket gemodelleerd kan worden (zoals de *n uit m beslissing* in FLOWer), in het andere geval betekent het dat er in het geheel geen adequate ondersteuning is (zoals ondersteuning van *dynamische subprocessen* in de meeste pakketten). We komen hier bij de bespreking van de patronen in detail op terug. We stellen de gebruikte patronen in enkele gevallen ter discussie, aangezien ze in onze optiek onvoldoende aansluiten op de praktijk. Zo is bijvoorbeeld de *case handling* filosofie of de datagedreven modellering niet met de onderzochte patronen te toetsen.

We zullen alle patronen bespreken. Het is raadzaam [WPA] bij de hand te houden aangezien daarin informatie over de patronen en de toepassingen ervan wordt gegeven. We tonen dat FLOWer *alle* patronen aan, zij het dat enkele patronen niet *direct* ondersteund worden. Maar ook in die gevallen is het *runtime* gedrag (dat wil zeggen het gedrag voor de eindgebruiker) in het algemeen exact gelijk aan hetgeen voorgeschreven is in de patronen. Voor de niet *direct* ondersteunde patronen biedt FLOWer eenvoudig te modelleren alternatieven aan, die geen ingewikkelde constructies vergen, zoals gebruik van API, uitstapjes naar Visual Basic of C++, gebruik van tellertjes of het bijhouden van een ingewikkelde administratie over de status van de zaak. Bovendien passen die constructies prima in de filosofie van FLOWer. Een kundige FLOWer modelleur zal de patronen direct toepassen. De expressiekracht ervan is vanzelfsprekend identiek aan hetgeen de patronen voorschrijven.

3 FLOWer model

Alvorens we in detail ingaan op de patronen, geven we eerst enige achtergrondinformatie over het FLOWer model. FLOWer is een *case handling* systeem, dat datagedreven werkt. De status van een zaak wordt bepaald

door de aanwezige informatie. De FLOWer Studio is de grafische ontwerpomgeving, waarin de *plan-elementen* (activiteiten, subplannen, beslissingen en milestones) in een flow-chart geplaatst worden. Hiermee kan de volgorde van afhandeling bepaald worden. Een pijl tussen twee *plan-elementen*, bijvoorbeeld van A naar B, betekent dat B pas kan worden uitgevoerd als A is uitgevoerd. Daarmee kunnen we een aantal patronen niet meer *direct* ondersteunen. Maar de besturing van FLOWer gaat veel verder en dat is een gevolg van het feit dat FLOWer datagedreven is. Het is in FLOWer bijvoorbeeld mogelijk om alvast informatie behorend bij een activiteit, die nog niet aan de beurt is, in te vullen of om informatie behorend bij een activiteit, die al aan de beurt is geweest, te wijzigen, zonder dat deze activiteit weer *expliciet* moet worden uitgevoerd. Deze vorm van besturing biedt veel extra flexibiliteit en wordt gemodelleerd door relaties te leggen tussen activiteiten (meer algemeen de plan-elementen) en data-elementen. Deze relaties zijn prima bruikbaar om enkele van de patronen te modelleren. Maar de mogelijkheden zijn te uitgebreid om deze relaties op een gebruikersvriendelijke manier in de flow-chart weer te geven. Dat zou tot een enorme hoeveelheid extra pijlen leiden. De consequentie daarvan is helaas wel dat het niet *direct* zichtbaar is in de afbeelding van het proces, zoals het onderzoek voorschrijft. Daardoor krijgt FLOWer niet overal een positieve beoordeling (+), hoewel de expressiekracht ruimschoots voorhanden is met de aanwezige, standaard door het model ondersteunde, constructies. Zie voor details [FPP]. Zie voor de voordelen van de datagedreven aanpak [GRP].

4 De patronen

In dit hoofdstuk behandelen we de twintig patronen uit [WPA]. Per patroon geven we de resultaten van het onderzoek, gevolgd door ons commentaar. De patronen zijn in de vorm van FLOWer procesontwerpen te vinden op de Pallas Athena website www.pallas-athena.com.

4.1 Pattern 1. Sequence

Score [WPA]:

+ *Directly supported through arcs connecting plan elements.*

De sequentie van activiteiten wordt door FLOWer volledig ondersteund. Een activiteit B kan pas worden gestart indien zijn voorganger A is afgerond. FLOWer kent echter nog een fijnkorreligere vorm van besturing: het is mogelijk om de bij B behorende data al ten dele in te voeren. Dit biedt additionele flexibiliteit, bijvoorbeeld als de gegevens al bekend zijn. Vanzelfsprekend is het ook mogelijk om te modelleren dat bepaalde gegevens alleen mogen worden ingevoerd indien B ook echt aan de beurt is. Dit om bijvoorbeeld te voorkomen dat de goedkeuring van een uitkering al gedaan kan worden voordat alle voorbereidende activiteiten volledig zijn afgerond. De genoemde functionaliteit vormt slechts een deel van het datagedreven model dat aan FLOWer ten grondslag ligt.

4.2 Pattern 2. Parallel split

Score [WPA]:

+ *Nodes in a subplan (static, dynamic and sequential) have an AND split semantics*

Ook dit patroon wordt door FLOWer volledig ondersteund. Met de hiervoor reeds genoemde datagedreven aanpak kan men de nodige flexibiliteit eenvoudig modelleren.

4.3 Pattern 3. Synchronisation

Score [WPA]:

+ *Nodes in a subplan (static, dynamic and sequential) have an AND join semantics*

Ook dit patroon wordt volledig ondersteund. Met de hiervoor reeds genoemde datagedreven aanpak kan men ook hier de nodige flexibiliteit modelleren.

4.4 Pattern 4. Exclusive Choice

Score [WPA]:

+ *Supported through the plan type system decision (based on data) and the plan type user decision (based on a user selection on the wavefront)*

FLOWer ondersteunt dit patroon volledig. Van belang hierbij is eveneens dat de keuze en erop volgende activiteit als één handeling uitgevoerd kan worden. Dit is van belang voor de ondersteuning van het *deferred choice* patroon, zie verderop patroon 16. Beslissingen worden in FLOWer gerepresenteerd middels het plantype decision. De modellering van de inhoud van de beslissing vindt binnen het plantype plaats. In feite wordt zo een blokstructuur aangemaakt, waarbij per uitkomst van de beslissing de dan uit te voeren activiteiten (en andere plan-elementen) worden gedefinieerd. Uiteindelijk synchroniseren de verschillende takken (één per uitkomst) weer, waarmee ook het volgende patroon volledig ondersteund wordt.

4.5 Pattern 5. Simple merge

Score [WPA]:

+ Supported by the end nodes of the plan type system decision and the plan type user decision
FLOWer ondersteunt dit patroon volledig.

4.6 Pattern 6. Multi-Choice

Score [WPA]

- Not supported. The decision plan types only allow for a 1-out-of-m decision.

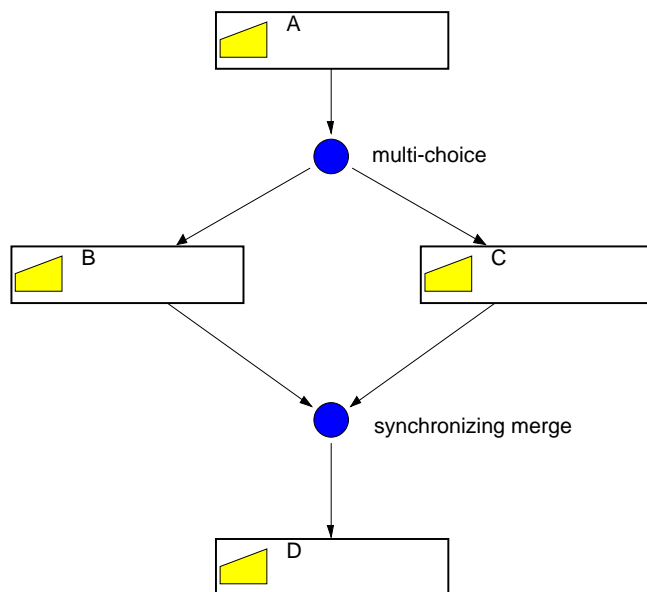
FLOWer ondersteunt dit patroon niet direct. Een *n uit m beslissing* kan gezien worden als *m separate 1 uit 2 beslissingen*. In figuur 3 hebben we een dergelijke constructie beschreven voor een *n uit 2 beslissing*. Die representeren we in FLOWer als 2 keer een *1 uit 2 beslissing*. De ondersteuning is wat betreft expressiekracht volkomen gelijkwaardig aan de *n uit m beslissing*. Slechts de *directe* representatie in de flow-chart ontbreekt. Dat verklaart de negatieve score. Daarnaast is de representatie van een *n uit m beslissing* in FLOWer voor de eindgebruiker duidelijk. Met andere woorden, in FLOWer is de expressiekracht aanwezig en het gebruik ervan is eenduidig.

4.7 Pattern 7. Synchronizing merge

Score [WPA]

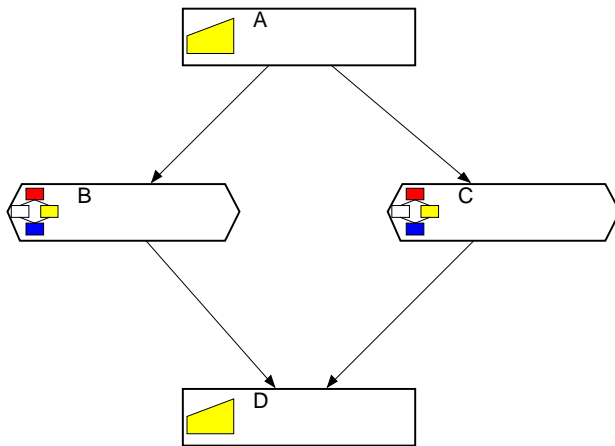
- Not supported

Helaas ontbreekt de uitleg in [WPA] bij de score van FLOWer. De constructie staat weergegeven in figuur 2.



figuur 2. Synchronizing merge

De mogelijke paden zijn: A, ABD, ACD, ABCD en ACBD. In FLOWer hebben we deze constructie gemodelleerd als aangegeven in figuur 3. De twee subplannen B en C zijn beslissingen. In beslissing B wordt hetzij activiteit B uitgevoerd, hetzij een lege tak doorlopen. In C net zo. Door deze vorm gedraagt FLOWer zich exact als voorgeschreven. De na de multi-choice gemodelleerde constructie hebben we ook voor patroon 6 gebruikt en wellicht dat daardoor de score laag is gebleken. Feit is echter dat de constructie uit figuur 3 een standaard FLOWer constructie is en bovendien het gedrag voor de eindgebruiker exact zoals voorgeschreven is. In onze optiek kan FLOWer het patroon aan.



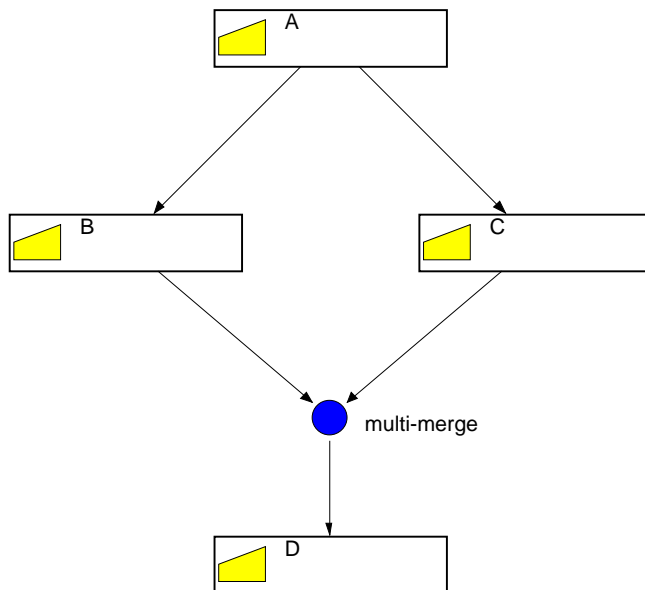
figuur 3. Synchronizing merge in FLOWer

4.8 Pattern 8. Multi-merge

Score [WPA]:

+/- It is possible to have multiple concurrent threads using dynamic subplans. Therefore, there is partial support for the pattern. However, since all networks are highly structured, it is not possible to have an AND-split/XOR-join type of situation.

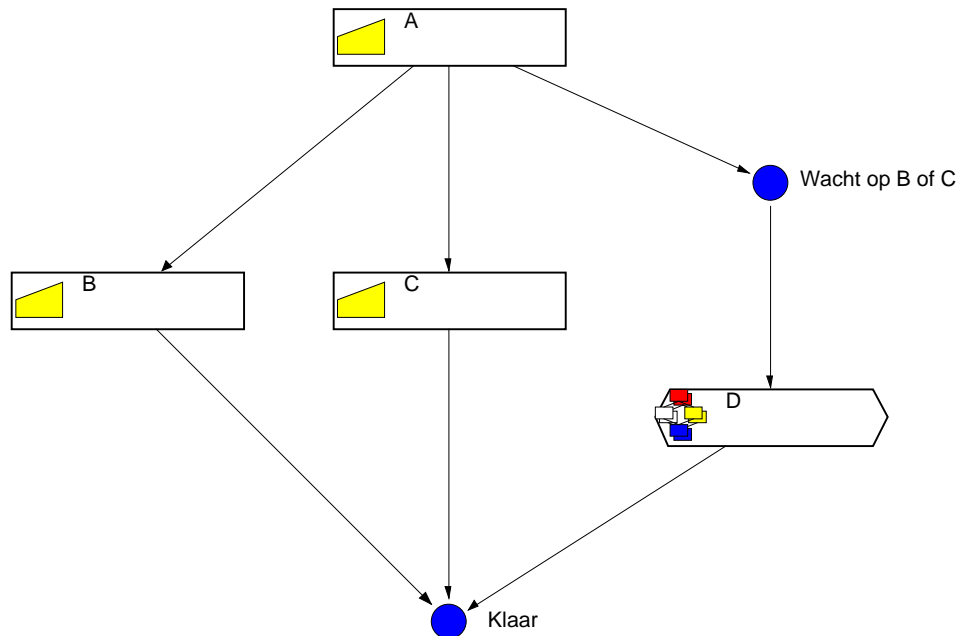
In dit geval zijn we het oneens met de score, mede doordat we het patroon ter discussie stellen. Het patroon (zie figuur 4) vereist dat delen van een proces meermalen kunnen worden uitgevoerd, te bepalen door het aantal samenkomende takken.



figuur 4. Multi-merge

In onze optiek is het in dat geval noodzakelijk dat het workflow management systeem onderscheid kan maken tussen de verschillende aldus ontstane instanties. In de praktijk is gebleken dat ook voor een gebruiker dit onderscheid per instantie gemaakt moet worden. Stel dat bijvoorbeeld zo'n procesdeel het afhandelen van een getuige betreft. Het is dan voor de behandelaar cruciaal dat het workflow management systeem onderscheid maakt (zonder ingewikkeld programmeerwerk) tussen getuige Jansen en getuige Pietersen. In FLOWer hebben we dat gerealiseerd door het dynamische subplan. In onze optiek dient dit patroon uitgebreid te worden en te toetsen op de aanwezigheid van dynamische subplannen (*multiple instances*) volgend op de multi-merge. We stellen dat we het in de praktijk relevante patroon volledig ondersteunen.

De opmerking over het highly structured zijn van de FLOWer networks is juist. Dat wil zeggen, de pijlen dwingen volgorde af. Maar zoals eerder toegelicht in hoofdstuk 3, kent FLOWer ook een fijnkorrelige vorm van besturing op basis van data. Daarmee kan dit patroon volledig gerealiseerd worden. In figuur 5 staat aangegeven hoe we het aanbieden in FLOWer. De milestone (“Wacht op B of C”) levert de benodigde semi-parallelliteit. Er kan pas een instantie van D (gemodelleerd als een zogenaamde dynamisch subplan) worden gemaakt nadat B of C is uitgevoerd. Bovendien wordt er een instantie van D gemaakt zodra B of C is uitgevoerd. Met andere woorden: er zijn precies op het juiste moment de juiste instanties van D beschikbaar en deze zijn terug te voeren op B of C.



figuur 5. Multi-merge in FLOWer

4.9 Pattern 9. Discriminator

Score [WPA]:

+/- The discriminator is not directly supported. However, dynamic subplans can have a so-called auto complete condition. This condition allows for emulating constructs like the discriminator and the n-out-of-m join.

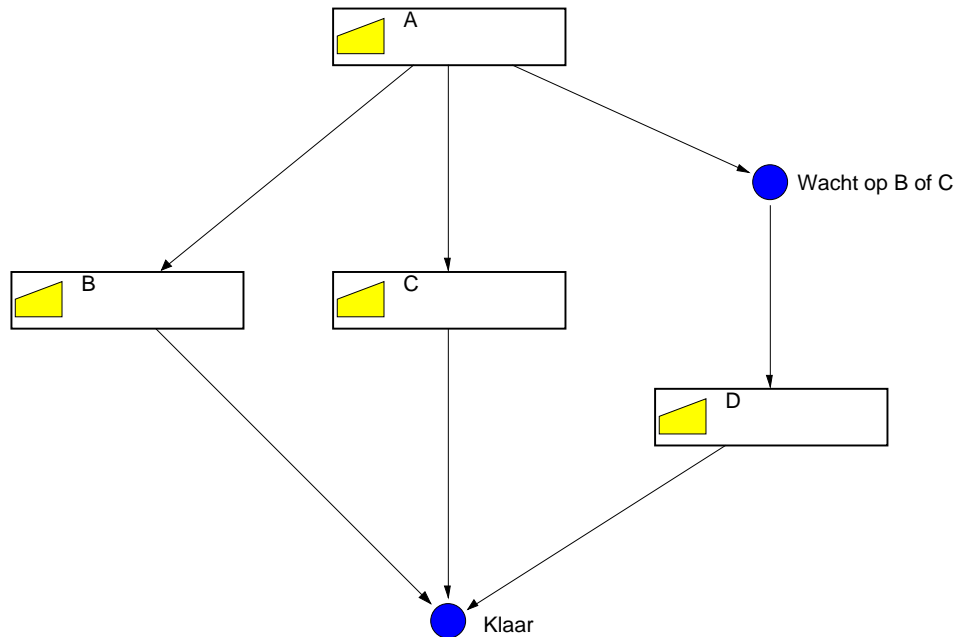
Een grafische weergave van dit patroon is lastig. Daarom de tekst uit [WPA]:

The discriminator is a point in a workflow process that waits for one of the incoming branches to complete before activating the subsequent activity. From that moment on it waits for all remaining branches to complete and “ignores” them. Once all incoming branches have been triggered, it resets itself so that it can be triggered again (which is important otherwise it could not really be used in the context of a loop).

FLOWer ondersteunt dit patroon niet direct. In feite is hier sprake van een semi-parallelliteit: D mag gedaan worden nadat B of C gedaan is. In FLOWer betekent een pijl tussen A en B dat B pas mag worden uitgevoerd nadat A is uitgevoerd. Dat wil zeggen: B wordt getoond op de statuslijn (of wavefront) nadat A is uitgevoerd. Door deze semantiek en het ontbreken van een “synchronizing merge” object zal FLOWer voor de wetenschappelijke definitie in [WPA] nooit een + of zelfs een +/- krijgen. Toch is er een eenvoudige constructie in FLOWer met exact dezelfde expressiekracht en exact hetzelfde gedrag voor de eindgebruiker. Met andere woorden, de niet aanwezige wetenschappelijke ondersteuning van het patroon wordt volledig opgevangen met een standaard FLOWer model. We gaan hier wat dieper op in. Allereerst kan in bovenstaand voorbeeld de informatie behorend bij B wel degelijk al worden ingevoerd. Dit is een van de eigenschappen van het onderliggende datagedreven model van FLOWer. Slechts de data-elementen die bij B als “restricted” zijn aangegeven, mogen pas worden ingevuld als B “aan de beurt” is. Daarmee is al veel van de gewenste semi-parallelliteit ondersteund, sterker nog, er is meer flexibiliteit aanwezig dan de semi-parallelliteit van het patroon. Die werkt immers op het niveau van activiteiten, hetgeen een veel grovere vorm van besturing is dan besturing op het niveau van aanwezige informatie (d.w.z. data-elementen).

Maar ook zonder gebruik te maken van deze flexibiliteit kan FLOWer het patroon aan. Zie figuur 6 voor de modellering in FLOWer. Na A plaatsen we de activiteiten B, C en D parallel. Door D vooraf te laten gaan door

een FLOWer *milestone*, die pas gepasseerd kan worden als B of C is uitgevoerd, ontstaat de gevraagde semiparalliteit. We merken op dat FLOWer *milestone* **zonder** conditie geen semantiek heeft en dus niet hetzelfde is als een toestand in een petri-net. Door een *milestone* echter te voorzien van een conditie, kunnen we in FLOWer de semantiek van een toestand wel modelleren.



figuur 6. Discriminator in FLOWer

4.10 Pattern 10. Arbitrary cycles

Score [WPA]:

- *Not supported. In fact there are no loops and the language is block structured with AND-type of blocks and XOR-type of blocks. Iteration is achieved through the sequential subplan and the redo role.*

FLOWer kent inderdaad geen *loops* in de zin van het onderzoek, dat wil zeggen dat je geen *loops* kunt tekenen in de modelleromgeving. Wel kent FLOWer dankzij het rollenmodel (zie [FPP]) de mogelijkheid om gebruikers flexibel delen van het proces opnieuw te laten uitvoeren. Afhankelijk van de rol van een gebruiker en de status kan een zaak teruggezet worden. Zo ontstaat er een groot aantal *loops* die anders stuk voor stuk getekend hadden moeten worden. Dat is vanzelfsprekend ondoenlijk. FLOWer is door dit model ongekend krachtig, maar biedt niet de mogelijkheid om *loops* te tekenen. Volgens de definitie gehanteerd in [WPA] krijgt FLOWer hier inderdaad een – als score, maar we zijn van mening dat het patroon niet voldoende aansluit op de praktijk. Uit de praktijk blijkt namelijk dat in een zaak geregeld door de gebruiker activiteiten opnieuw moeten worden uitgevoerd. Bovendien is de rol van de gebruiker bepalend voor hetgeen hij mag. Dit alles vastleggen in een flow-chart middels *loops* leidt tot hopeloos gecompliceerde modellen. We tekenen hierbij aan dat het vanzelfsprekend ook mogelijk is om de flexibiliteit te beperken. In FLOWer kan men eenvoudig zogenaamde *points-of-no-return* modelleren: zodra deze gepasseerd zijn, is het niet meer mogelijk terug te keren naar een activiteit vòòr dat *point-of-no-return*.

Daarnaast kent FLOWer door middel van het sequentiële subplan de mogelijkheid om een deel van het proces automatisch nogmaals uit te voeren, waarbij voor dat deel een nieuwe instantie ontstaat. We tekenen daarbij aan dat deze constructie dus anders is dan een *redo* waarbij een deel van het proces opnieuw doorlopen wordt (en de resultaten overschreven worden) en er maar 1 instantie is.

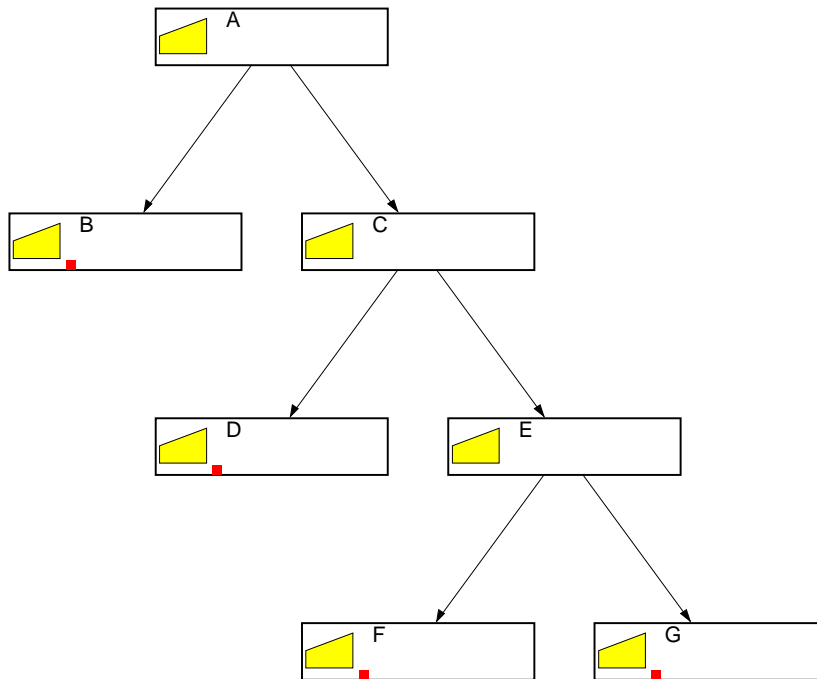
4.11 Pattern 11. Implicit termination

Score [WPA]:

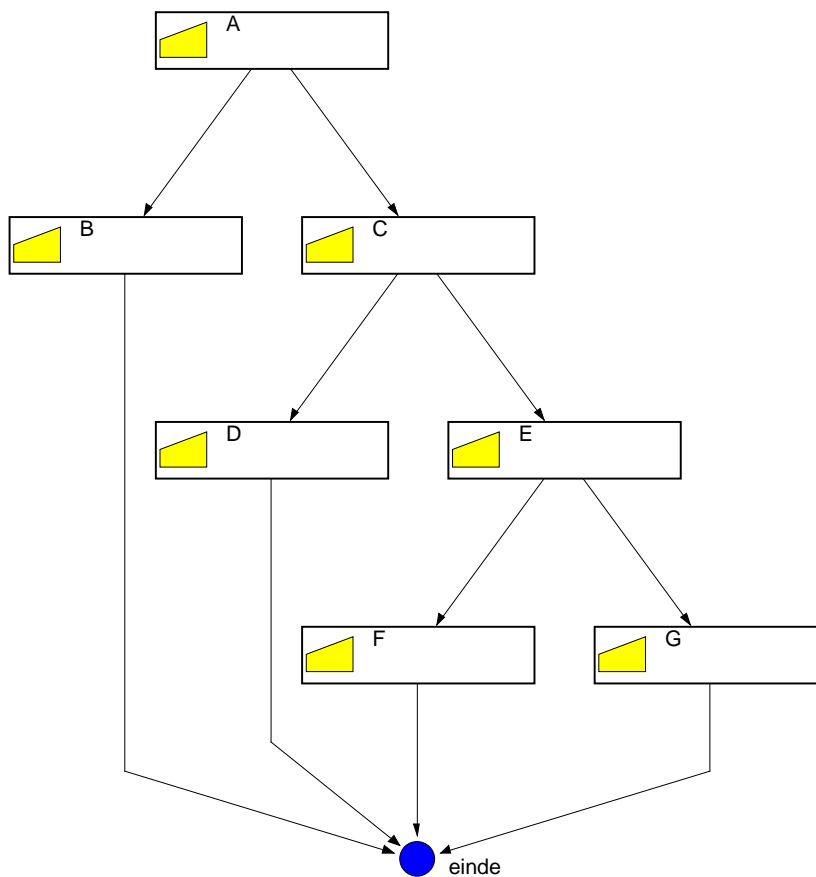
- *Not supported. Every plan has a unique begin and end node*

De score voor dit patroon valt teleurstellend laag uit. FLOWer eist dat de modelleur ieder plan voorziet van een unieke begin en einde node. Voor decisions is dit al automatisch gewaarborgd (dankzij de blokstructuur van FLOWer). Bij gebruik van AND splits (parallele paden) eist FLOWer niet dat deze allemaal met een

corresponderende AND-Join moeten worden afgesloten. Het is slechts nodig één enkele afsluitingsnode te tekenen. Dit vergt van de modelleur een zeer beperkte inspanning. We zijn derhalve van mening dat de aanwezige functionaliteit tenminste een +/- score zou moeten opleveren. In figuur 7 staat dit patroon weergegeven met behulp van eindactiviteiten (de activiteiten B, D, F en G, voorzien van een eindmarker). In figuur 8 staat het patroon weergegeven in FLOWer.



figuur 7. Patroon met eindmarkers.



figuur 8. Patroon met een enkele end milestone.

4.12 Pattern 12. Multiple instances without synchronization

Score [WPA]:

+ *Directly supported through dynamic subplans.*

Het dynamische subplan in FLOWer biedt alle voor de patronen 12 t/m 15 benodigde functionaliteit. De instelmogelijkheden zijn groot: zowel tijdens de modelleerfase als tijdens de afhandeling kan bepaald worden hoe vaak en onder welke voorwaarden een instantie gecreëerd moet worden. Dynamische subplannen zijn cruciaal voor de meeste werkprocessen. Overal waar procesdelen meermalen doorlopen moeten kunnen worden, is een dynamisch subplan nodig. Denk aan processen met klantinteractie, aan het afhandelen van getuigen, aan het behandelen van bezwaren (met een onvoorspelbaar aantal iteraties), het verwerken van uitval etc. Ook bij de geautomatiseerde procesdelen komt deze constructie uitstekend van pas. In de praktijk blijkt in het bijzonder patroon 15 cruciaal.

4.13 Pattern 13. Multiple instances with a priori design time knowledge

Score [WPA]:

+ *Directly supported through dynamic subplans and supported through a combination of splits and joins.*

Zie uitleg bij patroon 12.

4.14 Pattern 14. Multiple instances with a priori runtime knowledge

Score [WPA]:

+ *Directly supported through dynamic subplans. One can specify a variable number of instances.*

Zie uitleg bij patroon 12.

4.15 Pattern 15. Multiple instances without a priori runtime knowledge

Score [WPA]:

+ *Directly supported through dynamic subplans. It is possible to create new instances, while executing.*

Zie uitleg bij patroon 12. Patroon 15 kan bijvoorbeeld gebruikt worden bij de afhandeling van getuigen in een autoschadeprocess: pas tijdens de afhandeling is bekend of en hoeveel getuigen er zijn en van welke getuigen er verklaringen nodig zijn.

4.16 Pattern 16. Deferred choice

Score [WPA]:

+/- *There is no explicit notion of states. The plan type user decision (based on a user selection on the wavefront) can solve the implicit choice in some cases. The plan type system decision can solve the implicit choice in some other cases. Note that a system decision blocks until at least one of its conditions is true. This way "race conditions" based on time or external triggers are possible. In the latter case triggering is handled through data-dependencies rather than explicit control-flow dependencies. Moreover, mixtures of system and user decisions are problematic.*

Met een system of user decision is de deferred choice goed te modelleren. Er is in FLOWer geen sprake van een tijdsverschil tussen het nemen van de beslissing en het starten van de bijbehorende activiteit. Dit gebeurt instantaan. Overigens kan met een FLOWer milestone wel degelijk een toestand (*state*) gemodelleerd worden. Een milestone zonder conditie is inderdaad geen toestand (*state*), maar een milestone voorzien van de juiste *completion* conditie gedraagt zich exact als een toestand.

In onze optiek is daarmee de expressiekracht van FLOWer voldoende gewaarborgd. We merken ook nog op dat aan de gebruiker steeds de verschillende mogelijkheden (dat wil zeggen per *outcome* van de beslissing de eerste uit te voeren activiteit) aangeboden worden. De gebruiker (en/of het systeem, bijvoorbeeld in geval van een tijdtrigger of een ingekomen document) neemt impliciet de beslissing door een van die activiteiten uit te voeren. De start van de uitvoering van de activiteit leidt er automatisch toe dat instantaan alle andere alternatieven niet meer mogelijk zijn. Met nog andere woorden: men neemt niet expliciet een beslissing, maar men kiest voor de uitvoering van de eerste activiteit in een van de alternatieven.

FLOWer wijkt daarmee af van de traditionele workflow systemen waarin activiteiten in een werkbak worden geplaatst. FLOWer toont de beslissing en, in geval van een *deferred choice*, leidt de keuze voor een alternatief (dat wil zeggen de *start* van de uitvoering van de ene activiteit) dat instantaan alle andere alternatieven (de andere activiteiten) niet meer mogelijk zijn.

4.17 Pattern 17. Interleaved parallel routing

Score [WPA]:

+/- Due to the case metaphor there is just one actor working on a case. Therefore, there is no true concurrency and any parallel routing is interleaved. Since true concurrency is not possible, an intermediate rating is given.

Geen opmerkingen.

4.18 Pattern 18. Milestone

Score [WPA]:

+/- There is no direct support for milestones since there is no notion of states. However in all situations data dependencies can be used to emulate the construct. Simply introduce for each state (i.e. place in Petri-net terms) a data element.

Indedaad biedt FLOWer hier geen directe (in de tekening zichtbare) ondersteuning, maar de expressiekracht is daar niet minder om en derhalve is dit patroon goed ondersteund op de manier zoals aangegeven in bovenstaande tekst. In onze optiek biedt het sequentiële subplan in FLOWer daarbij een zeer adequate oplossing. Het patroon vereist dat een deel van het proces (B) eventueel meermalen (in sequentie) moet worden uitgevoerd. De uitvoering van B mag echter pas starten nadat A is uitgevoerd. Verder is het zo dat zodra een instantie van B succesvol is uitgevoerd, C uitgevoerd kan worden. Zodra C is uitgevoerd zijn er geen nieuwe instanties van B meer mogelijk. Met het sequentiële subplan is dit exact mogelijk. Bovendien garandeert het sequentiële subplan dat de verschillende instanties van B ook goed geadmineistreerd worden. Voor een gebruiker is dat eveneens van belang: die moet weten hoe vaak, op welke wijze en door wie B is uitgevoerd. In onze optiek is er pas sprake van directe ondersteuning indien er expliciet een sequentieel subplan gemodelleerd kan worden. In het patroon wordt hier echter geen rekening mee gehouden. Derhalve stellen we hier de definitie van *direct* ter discussie en daarmee de bepaling van de score. Functioneel biedt FLOWer in elk geval wat nodig is (d.w.z. onderscheiden van de verschillende instanties van B en het afhandelen van de juiste volgorde).

4.19 Pattern 19. Cancel activity

Score [WPA]:

+/- It is possible to skip or redo activities. However, it is not possible to withdraw an activity in one branch triggered by an activity in another branch. Skip and redo are explicit user actions. Therefore, they provide only partial support.

De opmerkingen zijn op zich correct. In onze optiek biedt de skip functionaliteit voldoende mogelijkheden, te meer daar men in FLOWer de skip autorisatie bepaalt door een *rol* en deze bovendien ook nog afhankelijk is van de status van een zaak. We merken op dat daardoor een zeer fijnkorrelige besturing van het *annuleren* van een activiteit ontstaat, die niet in een plaatje is weer te geven. Het wekt dan ook geen verbazing dat de ondersteuning in de praktijk meer dan voldoende is.

In onze optiek dient dit patroon verfijnd te worden teneinde de behoefte uit de praktijk, te weten een veel preciezere vorm van besturing afhankelijk van rol en status, weer te geven.

4.20 Pattern 20. Cancel case

Score [WPA]:

+/- It is possible to skip or redo an entire plan. However, skip and redo actions are always explicit user actions. Therefore, they provide only partial support. Note that using a data element named cancel and using this data element as a precondition for every activity in the flow it is possible to block a case. Although this is an elegant solution, it is still considered to be indirect.

In onze optiek is dit patroon niet goed te representeren in de flow-chart. Cancel case is een operatie op de *case*, de zaak, en dient derhalve als een te zetten attribuut voor de case gezien te worden. Een dergelijke aanpak is ook terug te vinden in het voorbeeld autoschadeproces waar een case in de status *escalatie* komt. Op dit moment is dat eenvoudig te modelleren met een data-element, zoals ook aangegeven in het onderzoek. Ondersteuning van het patroon is daarmee gewaarborgd.

5 Conclusies

Uit hoofdstuk 4 blijkt dat FLOWer de expressiekracht bezit benodigd om *alle* patronen te ondersteunen. Volgens de wetenschappelijke, *directe*, benadering scoort FLOWer zondermeer goed. Uit het bovenstaande blijkt echter dat alle patronen adequaat met direct in FLOWer aanwezige constructs kunnen worden opgevangen. FLOWer is daarin uniek, zoals ook blijkt uit de score van de andere pakketten op bijvoorbeeld de patronen voor dynamische subplannen.

Als we de definitie van directe ondersteuning vervangen door ondersteuning door middel van in het FLOWer model aanwezige *constructs*, dat wil zeggen zonder gebruik te maken van uitstapjes, API, extra programmeerwerk, dan komen we tot de volgende score:

patroon	score volgens <i>directe</i> ondersteuning	score volgens expressiekracht met standaard FLOWer
1 sequence	+	+
2 parallel split	+	+
3 synchronization	+	+
4 exclusive choice	+	+
5 simple merge	+	+
6 multi-choice	-	+
7 synchronizing merge	-	+
8 multi-merge *)	+/-	+
9 discriminator	+/-	+
10 arbitrary cycles *)	-	+/-
11 implicit termination	-	+/-
12 multiple instances without synchronization	+	+
13 multiple instances with a priori design time knowledge	+	+
14 multiple instances with a priori runtime knowledge	+	+
15 multiple instances without a priori runtime knowledge	+	+
16 deferred choice	+/-	+
17 interleaved parallel routing	+/-	+/-
18 milestone *)	+/-	+
19 cancel activity	+/-	+/-
20 cancel case *)	+/-	+/-

De met *) gemarkeerde patronen stellen we ter discussie. Meer algemeen concluderen we dat de patronen op dit moment geen mogelijkheden bieden om de fijnkorrelige besturing te toetsen, die naar onze mening in de meeste praktijksituaties nodig is voor een succesvolle toepassing van workflow management. Uitbreiding van de patronen voor toetsing van datagedrevenheid is wenselijk. Dat alles neemt niet weg dat de beschreven patronen een zeer waardevol middel zijn bij de beoordeling van workflow management systemen. Een middel dat al verder gaat dan de meeste bestaande mechanismen. Dat wat betreft de expressiekracht van de procesmodelleromgeving. Uit hoofdstuk 2 blijkt dat het onderzoek naar de patronen een belangrijk deel van de benodigde functionaliteit van workflow management systemen toetst, maar niet volledig is. Ondersteuning van de patronen is noodzakelijk, maar niet voldoende voor een succesvolle toepassing. De andere onderdelen van de in hoofdstuk 2 besproken cirkel voor procesbesturing dienen *ook* in patronen vervat te worden. Daarmee krijgt de potentiële gebruiker een toetsingsmechanisme in handen om tot een goede selectie te komen. Pallas Athena is gaarne bereid ook voor de andere onderdelen de uitdaging, gesteld door de wetenschappelijke wereld, aan te gaan. Juist op die onderdelen heeft Pallas Athena zich veel moeite getroost om een praktijkgerichte oplossing te implementeren en zal ook daar de kracht van FLOWer naar voren komen.

6 Literatuur

- [FPP] Case Handling met FLOWer: verder dan workflow. FLOWer positioneringsdocument, Pallas Athena, 2001.
- [GRP] Beyond Workflow Management: Product-Driven Case Handling. W.M.P. van der Aalst e.a., In S. Ellis, T. Rodden and I. Zigurs, editors, International ACM SIGGROUP Conference on Supporting Group Work (GROUP 2001), pages 42-51, ACM Press, New York, 2001.
- [WPA] Workflow Patterns, W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski and A.P. Barros, Eindhoven University of Technology e.a. 2002.