

PROCESGERICHT
SYSTEEMONTWERP

door

Eric Baardman & Stef Joosten

INHOUDSOPGAVE

Samenvatting	1
Inleiding	1
Processen vs. Systemen.....	2
Oplossing.....	5
Een architectuur voor procesgericht systeemontwerp.....	6
Ontwerpprincipes.....	12
Resultaten.....	14
Conclusies.....	15
Literatuur	15

Samenvatting

Business processen, die direct af te beelden zijn op software, helpen de communicatie tussen “business” en “ICT” te verbeteren. Om dit mogelijk te maken heeft Ordina een onderzoek uitgevoerd, waarin begrippen uit de proceskunde zijn afgebeeld op ontwerpconcepten uit de Unified Modeling Language (UML). Uit dit onderzoek kwam naar voren dat use-cases (een begrip uit de UML) een veroorzaker zijn van communicatieproblemen tussen procesontwerpers en systeemontwerpers. Dit probleem bleek te voorkomen door toepassing van een aantal specifieke principes bij het modelleren in de UML. Een modelleur ervaart het toepassen van deze ontwerpprincipes als “procesgericht systeemontwerp”. Het resultaat is gevalideerd door vast te stellen (met behulp van metamodellering) dat alle relevante proceskundige begrippen eenduidig kunnen worden afgebeeld op bouwbare entiteiten.

Inleiding

Welke consequenties heeft het procesdenken voor het ontwerpen van informatiesystemen? Kun je bestaande systemen niet gewoon aanvullen met een workflow engine (een “procesmotor”)? Waarom leidt de aansluiting tussen processen en systemen eigenlijk tot communicatieproblemen tussen procesontwerpers en systeemontwerpers?

Een onderzoek van Ordina naar oorzaken van gebrekkige aansluiting tussen processen en systemen in verschillende projecten heeft geleid tot een architectuurstudie naar bouwbare procesmodellen. Daarvoor zijn proceskundige begrippen afgebeeld op bouwbare entiteiten in de context van Rational-Rose, een bekende op de UML [5] gebaseerde repository voor

ontwerpmodellen. Het onderzoek leverde verrassend genoeg een verklaring voor het verschijnsel dat systeemontwerpers soms de analyses van procesanalisten opnieuw moeten uitvoeren. Zulk “dubbel werk” veroorzaakt niet alleen uitloop en kostenoverschrijdingen in projecten, maar ook irritaties bij gebruikers, die hun verhaal aan verschillende analisten opnieuw moeten vertellen. Naast een verklaring voor dit veel voorkomende probleem heeft deze architectuurstudie een aantal principes blootgelegd, die wij hebben samengevat als “procesgericht ontwerpen van informatiesystemen” of kortweg “procesgericht systeemontwerp”. Toepassing van deze principes levert een merkbaar eenvoudiger systeem op dat beter aansluit op de werkwijze van gebruikers¹. Dit artikel bespreekt de resultaten van het uitgevoerde onderzoek.

Processen vs. Systemen

De aansluiting van processen en systemen is een bekend probleemgebied en wordt zelfs genoemd als oorzaak van het falen van sommige projecten. Voor ons onderzoek is een analyse gemaakt van een drietal projecten²:

Project A handelde om het ontwerp en de realisatie van een shared service centrum voor back-office werk in een grote financiële instelling. De totale projectomvang was ongeveer Eur 40 mln. Ordina deed mee in de realisatie, en was verantwoordelijk voor de integratie van de procesmotor (een case handling engine) in het ICT-gedeelte van het totale project. Bij binnenkomst is een analyse gedaan van de status van het ontwerp en een kwalitatieve conclusie getrokken dat de functionele kant en de procesbesturingskant van het ontwerp niet op elkaar aansloten. In het kader van ons onderzoek heeft een geautomatiseerde detectie van overtredingen van de samenhang tussen

¹ Procesgericht systeemontwerp is voor Ordina's procesarchitecten één van de basiscompetenties.

² De organisaties achter deze projecten en het bronmateriaal zijn bij Ordina bekend. De onderzoekers respecteren de anonimiteit van de onderzochte organisaties.

systeemmodellen en procesmodellen plaatsgevonden. Hieruit kwamen meer dan 5000 overtredingen aan het licht.

Project B handelde om het ontwerp van een kredietensysteem, waarbij Ordina verantwoordelijk was voor het proceskundige gedeelte van het systeem-ontwerp en blauwdruk voor de nieuwe organisatie heeft gemaakt. De projectomvang hiervan was ongeveer Eur 3 mln. Ook in dit project bestond spraakverwarring tussen proceskundigen en systeemkundigen. Als deel van ons onderzoek zijn metamodellen gemaakt die proceskundige begrippen en functionele begrippen op elkaar afbeelden. Vanuit deze analyse werd duidelijk dat het begrip use-case, dat in de UML gebruikt wordt om het werk van personen te beschrijven, vanuit de proceskunde gezien veel verschillende betekenissen kent.

Project C ging over de inrichting van een gedeeltelijk geautomatiseerde verzekerings-backoffice. De omvang van het project was ongeveer Eur 8 mln. Ordina was verantwoordelijk voor het gehele ontwerp, waarbij werkverdeling op basis van competenties van medewerkers was vereist (skill based routing). Hiervoor is een speciale procesmotor ontworpen. De metamodellen voor deze procesmotor zijn voor dit onderzoek bewerkt tot een referentiemodel, die competentie-gebaseerde werkverdeling generiek toepasbaar maakt in andere situaties.

Nadere bestudering van projectdocumentatie liet in alle gevallen spraakverwarring zien rond het begrip use-case. Conceptuele analyse³ leverde een verklaring: verschillende proceskundige begrippen worden allemaal als use-case behandeld. Het procesmatige onderscheid tussen verschillende abstractieniveaus gaat verloren wanneer elke soort werk als use-case wordt behandeld. Het Proces Architectuur Model (PAM) [3] maakt bijvoorbeeld onderscheid tussen procesniveau, procedureniveau, werkniveau en handlingenniveau, en gebruikt verschillende begrippen op de verschillende

³ Deze analyse is uitgevoerd in de formele analysetechniek CC, een dialect van de relationele algebra.

niveaus. Op procesniveau worden verschillende processen aan waardenketens gekoppeld, en worden per proces een aantal procedures benoemd. Eén slag concreter krijgen procedures inhoud en structuur door middel van fases (bijvoorbeeld adviseren, accepteren, offreren en inschrijven voor een hypotheek) en stappen (bijvoorbeeld “beoordelen krediet”, of “fatteren”). De taken, die een individuele medewerker krijgt toebedeeld bestaan uit een mogelijk groot aantal handelingen. Procesontwerpers gebruiken veel meer verschillende concepten (zoals proces, procedure, fase, stap, taak, handeling) dan alleen use-cases om bruikbare en bouwbare modellen te maken van processen in organisaties.

Methoden rond de UML, waarvan de Unified Process [4] de bekendste is, beginnen met een use-case analyse, waarin het werk van actoren wordt geanalyseerd door scenario's uit de praktijk te registreren. In recente literatuur wint het begrip “business use-case” aan populariteit, waarmee meestal een abstractieniveau hoger wordt bedoeld.

Een verklaring voor het gebrek aan aansluiting tussen processen en systemen is dat verschillende soorten werk in de UP onder één noemer, use-case (c.q. business use-case) worden gebracht. Een hypotheekaanvraagprocedure kan bijvoorbeeld in z'n geheel als use-case worden benoemd, maar ook een kleine handeling zoals het versturen van een ontvangstbevestiging is een use-case. Vanuit de Unified Process gezien is dat geheel volgens het boekje. Vanuit de business geredeneerd gaat daarmee echter belangrijke informatie uit de procesanalyse verloren. Use-cases zijn (ook volgens het boekje) immers bedoeld om de gebruikersinteractie te beschrijven met een informatiesysteem. Het beschrijven van een proces gaat veel verder: werkverdeling, autorisaties, rolverdeling, verantwoordelijkheden, enzovoorts. Ziedaar een bron van ergernissen en conflicten: de UML-modelleur werkt keurig volgens het boekje en de procesmodelleur ziet kostbare informatie verloren gaan.

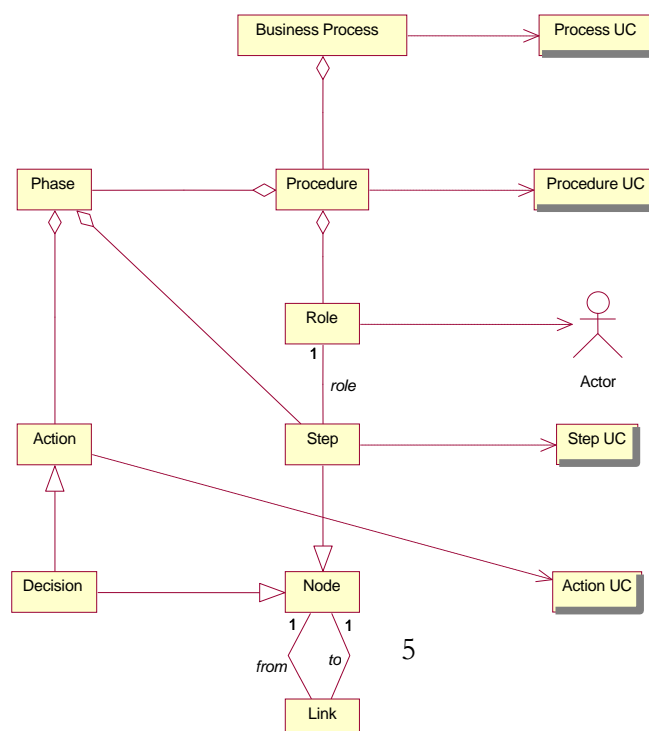
Oplossing

Een oplossing voor dit vraagstuk is gezocht in de wijze van werken binnen de grenzen van de bestaande methoden en technieken (dus PAM voor de processen en UML voor de systemen). De UML kent stereotypes, waarmee het onderscheid tussen verschillende soorten werk kan worden weergegeven als verschillende soorten use-cases. Vanuit de probleemanalyse is de volgende hypothese geformuleerd:

Aansluiting tussen procesmodellen en systeemmodellen in de UML kan worden verbeterd door verschillende typen use-case te onderscheiden voor werk op verschillende abstractieniveaus.

De feitelijke afstemming heeft plaatsgevonden door het maken van metamodelen. Het metamodel vanuit de processen (figuur 1) is gebaseerd op case handling volgens PAM [3]. Het getoonde metamodel laat links de essentie zien van de procesmodellen en rechts de afbeelding naar use-case stereotiepen, zoals die in het onderzoek zijn gebruikt.

Figuur 1: onderscheid tussen soorten use-case



Door meerdere soorten van use-case te onderscheiden wordt meer kennis vanuit de procesmodellen behouden. Zo worden bijvoorbeeld in een procedure-use-case de scenario's beschreven waarin een zaak (case) compleet wordt afgewikkeld op zijn "reis" door de organisatie. In de step-use-cases zijn scenario's beschreven waarbij een actor precies één taak afwikkelt, zodat het werk kan worden overgedragen naar de volgende stap in de procedure. In Rose zijn de verschillende typen use-cases weergegeven door verschillende diagrammen, op een manier die traceerbaarheid in stand houdt⁴.

In de praktijk leverde deze oplossing een hogere ontwerpdiscipline⁵ op, wat leidde tot eenvoudiger modellen. Een hypotheekaanvraagprocedure met 23 stappen was bijvoorbeeld na 3 maanden teruggebracht tot een procedure van 7 stappen. Onze ervaring suggereert dat deze reductie, ongeveer een factor 4, gebruikelijk is ofschoon er onvoldoende data is verzameld om harde uitspraken hierover te kunnen verantwoorden.

Een architectuur voor procesgericht systeemontwerp

Om te weten of de gevonden oplossing valide is, moeten we terug naar de oorspronkelijke taakstelling: het maken van bouwbare modellen. Daartoe is een architectuur gemaakt, waarin elk proceskundig concept direct op een bouwbare entiteit is afgebeeld. In deze architectuur heeft elke medewerker een "werkbak" (werkvoorraad) waarin alle relevante taken zichtbaar zijn. Neemt een medewerker een taak uit haar of zijn werkbak, dan bepaalt een procesmotor⁶ welke handelingen uitgevoerd kunnen worden door te kijken naar de onderliggende procedure, het autorisatie- c.q. competentieprofiel van de betreffende medewerker en de toestand van de onderhanden zaak. Op grond daarvan wordt een taakscherm-op-maat aangeboden, waarin de

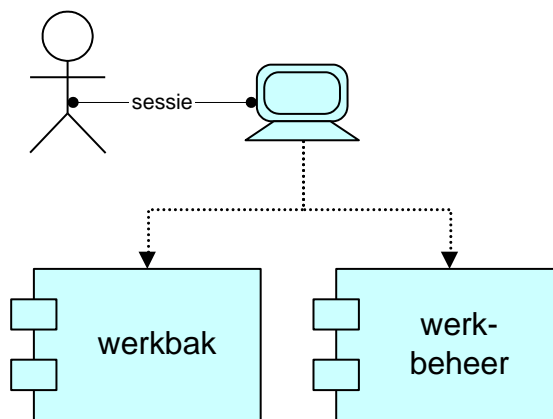
⁴ Om de traceerbaarheid in stand te houden zijn concessies gedaan aan het metamodel, wegens technische beperkingen van Rose.

⁵ Dit werd in het project met de term "strak ontwerpen" en "opstrakken" van een naam voorzien.

⁶ Meestal wordt voor de procesmotor een commercieel verkrijgbare BPM-engine gebruikt.

medewerker precies de informatie, de velden en de knoppen krijgt die nodig zijn voor deze specifieke taak⁷ in deze situatie.

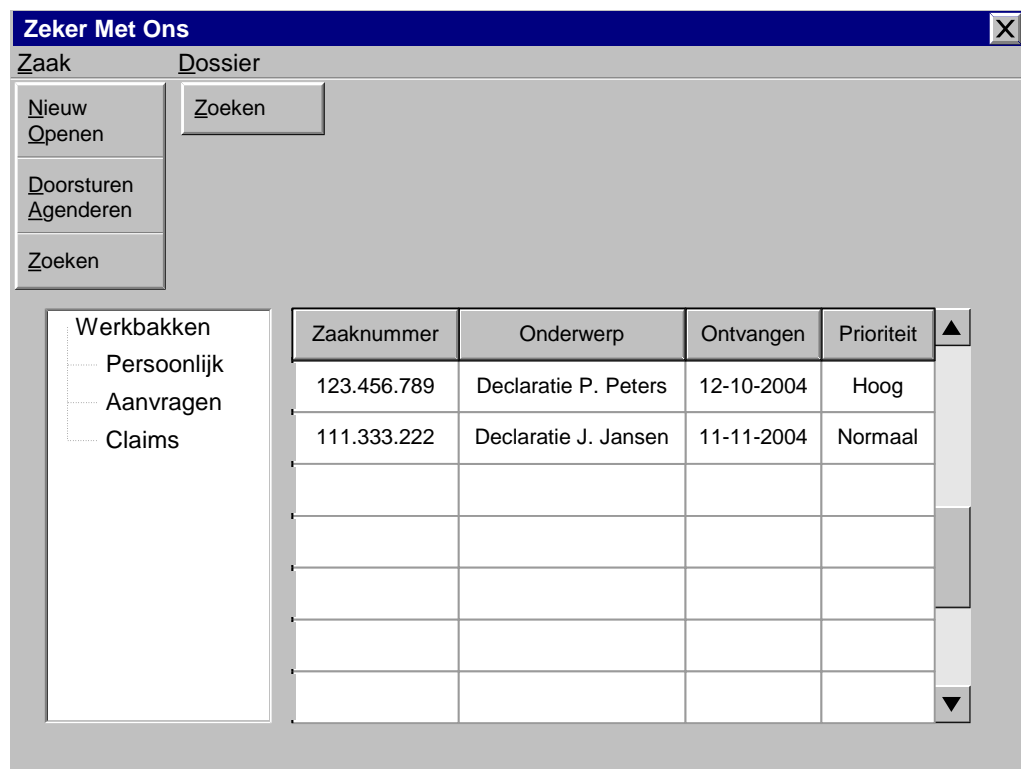
De client-applicatie (de applicatie die draait op het workstation van de gebruiker) kent een dialoogstructuur in twee niveaus: het werk-niveau en het taak-niveau.



Figuur 2: client op werkniveau.

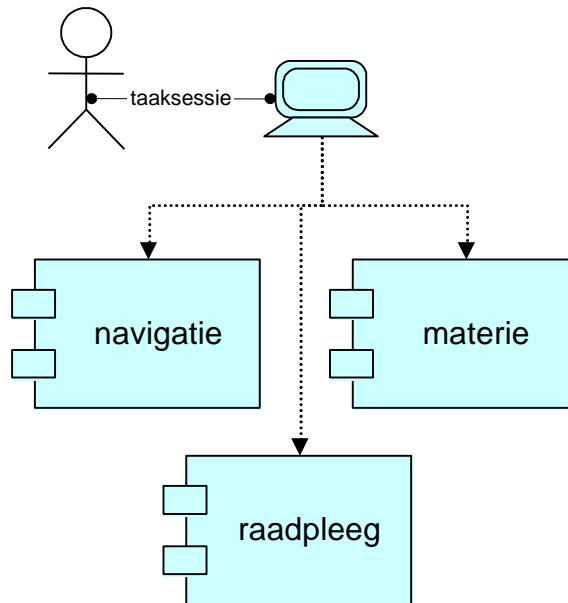
Het werk-niveau (Figuur 2) is het niveau waar de gebruiker zich direct na inloggen bevindt. Op dit niveau ziet de gebruiker een werkbak (werkvoorraad), waarin de taken zitten die op dat moment gedaan kunnen worden. Elke taak in de werkbak gaat over precies één zaak. Naast de werkbak biedt de client-applicatie functionaliteit waarmee de gebruiker zijn of haar eigen werk kan beheren, zoals taken overdragen, eigen productiviteit raadplegen, vakanties doorgeven, enzovoorts. In de client-applicatie is dit zichtbaar door een aantal knoppen voor taken, die niet onder besturing van de procesmotor worden aangeroepen. Figuur 3 geeft een voorbeeld van het werk-scherm, dat een gebruiker te zien krijgt.

⁷ Een zaak in een werkbak noemen we *taak*. Een medewerker ziet de werkbak dus als een lijst van taken.



Figuur 3: Voorbeeld scherm op werkniveau

Het aanklikken van een taak uit de werkbak of een taakknop opent het taakniveau (Figuur 4), waarin de gebruiker consequent te maken krijgt met drie deelschermen: 1) een navigatie-scherm om bij de relevante handelingen te komen, 2) een materie-scherm, met daarin de materiespecifieke informatie, invulvelden, functionele knoppen, enz. en 3) een raadpleegscherm, waarin onderliggende applicaties informatie kunnen verstrekken zoals klantbeeld, opgevraagde documenten en dergelijke.



Figuur 4: client op taakniveau

Het materie-scherm biedt de mogelijkheid om deelschermen uit andere applicaties in te voegen op een transparante wijze. Voor de gebruiker zijn deze applicaties dan niet als losse applicaties te onderscheiden; het geheel biedt een geïntegreerde, consequente werkomgeving. Omdat transacties alleen in het materiescherm plaatsvinden, is de consistentie tussen data in de onderliggende systemen en data in de client-applicatie een beheersbaar vraagstuk⁸. Figuur 5 geeft een voorbeeld van een taakscherm.

⁸ Ofschoon dit vraagstuk essentieel is voor een goede totaaloplossing, valt een bespreking ervan buiten het bereik van dit artikel.

Zaaknummer 123.456.789

Client informatie

Polisnummer: 20001010

Naam: P. Peters

Meer...

Handeling	Status
Invoeren declaratie	Gereed
Dekkingsbeoordeling	Uitvoeren

Dekkingsbeoordeling

Product: Polis Plus

Dekking: Geneesmiddelen

Eigen risico: € 500,-


Kosten dit jaar: € 1000,-

Percentage vergoeding: 75%

Declaratie: € 30,-

Vergoeding volgens polis: € 22,50

Uitbetalen Niet uitbetalen



ZMO
Zeker Met Ons Verzekeringen

Declaratieformulier

NAAM: D. Peters

ADRES: Kerkstraat 1

POSTCODE: 1234 AA

WOONPLAATS: Dorpsaan de Vind

DATUM	OMSCHRIJVING	BEDRAG (€)
10-09-2004	Dijnstillers	30,-

Figuur 5: Voorbeeld taakscherm

De geschetste architectuur maakt het werken voor een gebruiker zo eenvoudig mogelijk. Navigatie door een complexe menustructuur is overbodig, omdat de dynamisch samengestelde taakschermen precies de juiste functionaliteit aanbieden op basis van de onderhanden zaak, de procedure die daarop wordt uitgevoerd en de eigenschappen van de gebruiker. Een

gebruiker “navigeert” uitsluitend om een handeling te selecteren in de wetenschap dat elke aangeboden handeling is toegestaan en ook nuttig en nodig werk met betrekking tot de onderliggende zaak bevat. Het taakscherm laat slechts handelingen zien die de gebruiker op dat moment mag en kan doen. Alle informatie, die nodig is om een handeling te voltooien is direct vanuit het taakscherm toegankelijk. Wanneer er progressie in de onderhanden zaak wordt bereikt, zal automatisch binnen het taakscherm bepaald worden of er vervolg handelingen uitgevoerd kunnen worden door de gebruiker.

Consequentie is dat de navigator op een taakscherm dynamisch wordt opgebouwd. Dat sluit een statische menu-structuur uit. Dat is terug te zien in de technische architectuur, die een navigatie-component bevat om dynamisch een navigator samen te stellen.

Het client-ontwerp kan op basis van deze procesgerichte architectuur eenvoudiger zijn dan een integratie van een afzonderlijke workflow-client, document-client en materiesysteem-client. De verschillende componenten, waaruit het totale systeem is opgebouwd, moeten als één geïntegreerde applicatie bij de gebruiker terechtkomen.

concept	use case stereotype	bouwbare entiteit
proces	process use case	procescomponent
procedure	procedure use case	
processtap	step use case	
handeling	action use case	

Ontwerpprincipes

De belangrijkste voorwaarde voor een goed ontwerp is een adequaat besturingsmodel van de organisatie. Het “ontwerpen vanuit de business” is een lege kreet, wanneer niet duidelijk is hoe “de business” wordt bestuurd. Om bedrijfsdoelstellingen te behalen worden meet-, beslis- en stuurinstrumenten gebruikt. Hiermee wordt onder andere het bedrijfsproces als wel de organisatie gemeten en bestuurd. De wijze waarop is terug te vinden in het besturingsmodel.

Procesgericht systeemontwerp is gebaseerd op een aantal principes. Een eerste principe is om uitgaande van het besturingsmodel van de organisatie, eerst procedures op de juiste wijze af te bakenen. Pas dan kunnen gebruikers zinvol meepraten over scenario's, die een analist vervolgens kan vastleggen in use-cases. Foutieve afbakening van procedures kan leiden tot ingewikkelde use-cases, die ook nog eens overlap kunnen vertonen. In de realisatie vertaalt zich dat naar redundante en onlogische schermen, overvloedige informatie- en invoervelden en/of het doorlopen van onnodige schermen. Daarnaast helpt het besturingsmodel om de discussies over de gewenste situatie te focussen. Organisaties die weinig tot geen ervaring hebben met workflow en/of document management zijn moeilijk in staat om aan te geven op welke wijze ze in de toekomst willen/kunnen werken. De business case samen met het besturingsmodel geeft hierin houvast.

Een tweede principe is het ontkoppelen van organisatiemodel en procedurevoorschriften. In een organisatiemodel is geregeld welke personen welke functies bekleden en wat hun autorisaties zijn. In procedures wordt geregeld welke rollen welke handelingen mogen uitvoeren. Door rollen (vanuit de procedures) en functies (vanuit de organisaties) strikt te onderscheiden ontstaat ruimte om het organisatiemodel te ontkoppelen van procedures. Immers, als procedures afhankelijk zouden zijn van organisatorische functies, dan moeten bij elke reorganisatie of fusie alle procedures worden herijkt. In gemeentes, bijvoorbeeld, kent men meer dan

1000⁹ procedures. Ontkoppeling van procedurevoorschriften en organisatiemodel loont bijvoorbeeld ook bij reorganisatie en/of fusie. Daarnaast is deze ontkoppeling waardevol voor organisaties die te maken hebben met piekperiodes in hun processen. Teams kunnen eenvoudig *real-time* bijgeschakeld worden of een andere werkverdeling krijgen.

Een afgeleide van het voorgaande principe is het weglaten van activiteiten. Het modelleren van activiteiten in een procesmodel betekent dat de ontwerper op de tekentafel bepaalt langs welke route het werk door de organisatie stroomt [2]. Dit blokkeert een dynamische vorm van routeren, waarbij pas op het moment zelf bepaald wordt, aan de hand van de toestand van een zaak¹⁰ en de autorisatie profielen van medewerkers welke handelingen kunnen worden uitgevoerd. Dit verklaart waarom procesmodellen in voorkomende gevallen verstarrend werken.

En vierde principe is het gebruik van mijlpalen. Een mijlpaal in een proces is een “point of no return”. Als bijvoorbeeld een bank haar fiat geeft op een financieringsaanvraag, als een vergunning wordt verleend, of als de koper van een huis de koopovereenkomst sluit, ontstaat er voor de onderliggende processen een nieuwe situatie. Vóór het bereiken van een mijlpaal gebeurt ander werk dan erna. Het op je schreden terugkeren (rollback), wat in systeemontwerp gemeengoed is, is in de processen uit den boze. Het aantal mijlpalen in een procedure is veelal beperkt tot 2 à 3, ofschoon er geen grens op zit.

Een vijfde principe is het ontwerpen van bedrijfsservices die onafhankelijk zijn van:

- de procesinrichting, de service heeft en bewaart geen kennis over het proces;
- de locatie van uitvoering van het proces

⁹ De lijst van werkprocessen van de Vereniging Nederlandse Gemeenten (febr 2004) bevat er 1024.

¹⁰ Een *zaak* is een instantie van een procedure. Bijvoorbeeld: de huursubsidie-aanvraag van de familie de Vries is een zaak, die wordt afgewikkeld door de procedure “aanvragen huursubsidie” te volgen.

De bedrijfservice is een eenheid van transactie, consistentie en autorisatie. Het ontbreken van proceslogica in services, maakt services gemakkelijker herbruikbaar. Berekeningen zijn daarvan een goed voorbeeld. Renteberekeningen of bruto/netto berekeningen zijn alleen in verschillende situaties herbruikbaar wanneer zij zich tot hun “sometje” beperken. Dit bespaart dubblures in software inclusief de kosten om de dubblures consistent te houden.

De besproken principes hebben invloed op zowel de systemen als de processen. Daarom is procesgericht systeemontwerp meer dan de optelsom van systeemmodellen en procesmodellen. Het structurele verband tussen deze modellen is van belang. Meer modelleerprincipes zijn terug te vinden in bijvoorbeeld [3].

Resultaten

Een procesgericht informatiesysteem werkt voor een gebruiker in de praktijk eenvoudiger dan een klassiek ontworpen informatiesysteem. Dit heeft een aantal oorzaken:

- Navigatie naar taken ontbreekt, omdat de procesmotor dit uit handen van de gebruiker heeft genomen. In systemen zonder procesondersteuning navigeert de gebruiker zelf, wat soms complex is.
- Goed ontworpen processen leveren logische taken op. De gebruiker krijgt in het taak scherm consequent te maken met één taak, die zonder “ballast” wordt gepresenteerd.
- De services in de middleware bevatten geen proceslogica en zijn dus gemiddeld gesproken “klein”. Een primitieve service komt overeen met een transactionele eenheid op database niveau. Dit vereenvoudigt het ontwerp van services.

- Het principe van eenmalige invoer aan de bron is vanzelfsprekend. Dit bespaart veel invoer-dialogschermen.

Of procesgericht systeemontwerp goed wordt toegepast is te beoordelen aan het resultaat. Wanneer een project alleen maar groeit in complexiteit gaat het *niet* goed. In een procesgericht ontwerp gaan modellen “inklinken” en wordt het geheel eenvoudiger. In één van de onderzochte projecten trad een vereenvoudiging in een procesmodel (over hypotheekaanvragen) op van 24 processtappen naar 9 in een periode van 3 maanden tijd.

Conclusies

Proceskundige principes zijn van nut bij het ontwerpen van procesgerichte informatiesystemen. Het helpt dus niet alleen om organisaties te stroomlijnen, maar ook om de ondersteunende informatiesystemen eenvoudiger te maken. De resultaten in de praktijk zijn bemoedigend, en wijzen op toegenomen eenvoud en grotere beheersbaarheid in grootschalige trajecten.

Vervolgonderzoek zal moeten uitwijzen of deze principes in ontwerptools, zoals ARIS en Rational Rose, kunnen worden ingebouwd. Vooralsnog hangt het toepassen ervan in de praktijk af van goed opgeleide professionals.

Literatuur

- [1] Baardman, Eric *Oog voor de organisatie*, Business Process Magazine 6(7), pp. 34-41, oktober 2001.
- [2] Joosten, S. *Rekenen met taal*. Informatie 5(42) , pp. 26–31, mei 2000.
- [3] Joosten, Stef en anderen, *Praktijkboek voor Procesarchitecten*, Kon. van Gorcum, Assen, september 2002.
- [4] Kruchten, Ph., *The Rational Unified Process – An Introduction*, Addison-Wesley-Longman, Reading, Mass., 2000.
- [5] Rumbaugh, J., Jakobson, I., and Booch, G. *The Unified Modeling Language Reference Manual*. Addison-Wesley, Reading, Mass., 1999.