Software-ontwikkeling : beheersen en onzekerheid

Software-ontwikkeling; beheersen en onzekerheid

Fred J. Heemstra

Dat begroten en beheersen van software-ontwikkeling lastig is, behoeft nauwelijks toelichting. Berichten in de media over uit de hand gelopen automatiseringsprojecten bevestigen dit. De belangstelling voor 'kostenbeheerst' ontwikkelen van software neemt toe, getuige het groeiende aantal publikaties over dit onderwerp. Van pasklare oplossingen of aanzetten daartoe is in deze publikaties nauwelijks sprake. Veelal zijn de artikelen probleembeschrijvend van aard of inventarisaties van bestaande begrotingsmodellen en metrieken. Voor een ordening van de problematiek zijn deze publikaties waardevol; zij geven de software-ontwikkelaar evenwel geen directe ondersteuning bij de begroting en beheersing. In deze bijdrage wordt een aanzet gegeven om dit manco op te heffen. Uitgangspunt is dat elk software-ontwikkelproject vraagt om zijn specifieke vorm van beheersing en begroting. In dit artikel worden vier situaties, die bij software-ontwikkeling kunnen optreden, onderscheiden en wordt aangegeven welke vorm van beheersing en begroting het beste aansluit bij een bepaalde situatie. Voor een gedetailleerde uiteenzetting over dit onderwerp wordt verwezen naar Heemstra (1989).

1 Inleiding

Begroten en beheersen van software-ontwikkeling is moeilijk. Dit blijkt onder andere uit een onderzoek van Heemstra, Siskens en Van der Stelt (1989). De belangrijkste resultaten van dit onderzoek, waaraan 597 Nederlandse organisaties meewerken, kunnen als volgt worden samengevat:

- 35% van de organisaties die aan het onderzoek deelnamen, maakt geen begroting voor software-ontwikkeling;
- 50% van de deelnemende organisaties registreert niets van software-ontwikkelprojecten;
- 57% van de deelnemende organisaties voert geen nacalculaties uit van een software-ontwikkelproject;
- 80% van de projecten die door de deelnemende organisaties worden uitgevoerd, heeft te kampen met budget- en levertijdoverschrijdingen;
- de gemiddelde budget- en levertijdoverschrijding bedraagt 50%;
- 62% van de deelnemende organisaties die een begroting opstelt, doet dit op basis van intuïtie en ervaring. Slechts 16% maakt gebruik van formele begrotingsmethoden, te weten begrotingsmodellen.

Een uitgebreid onderzoek, uitgevoerd door de Universiteit van Arizona, komt voor de overschrijding van budgetten en doorlooptijden tot nagenoeg dezelfde resultaten (Phan, Vogel en Nunamaker, 1988). Ook in andere publikaties worden de problemen bij het begroten en beheersen van software-ontwikkeling met harde cijfers onderbouwd (van Genuchten en Fierst van Wijnandsbergen, 1989) (van Lierop en Volkers, 1989) Thambain en Wilemon, 1986).

Er is een groot aantal oorzaken te noemen waarom het moeilijk is de ontwikkeling van software te beheersen (Heemstra, 1989). Ik wil me in dit artikel beperken tot het signaleren van slechts één belangrijke oorzaak, namelijk de beperkte theorievorming over het begroten en beheersen van software-ontwikkeling. Hiermee komen we direct tot de kern van dit artikel: *het geven van een theoretisch raamwerk voor het begroten en beheersen van software-ontwikkeling.* Het uitgangspunt in dit raamwerk is de vooronderstelling dat er niet sprake kan zijn van een standaardaanpak om software-ontwikkeling te begroten en beheersen. Er zijn verschillende beheerssituaties te onderscheiden, waarbij iedere situatie om zijn specifieke aanpak van beheersen en begroten vraagt.

In paragraaf 2 wordt aangegeven dat de basis voor het onderscheid in verschillende beheerssituaties de onzekerheid is omtrent het produkt dat ontwikkeld moet worden, de wijze waarop het produkt ontwikkeld moet worden (het proces) en de middelen die daar voor nodig zijn. In paragraaf 3 worden vier beheerssituaties onderscheiden en in paragraaf 4 wordt voor elke situatie aangegeven welke beheersstrategie en wijze van begroten het best hierbij aansluit. Het artikel wordt afgesloten met conclusies (paragraaf 5).

2 Onzekerheid en beheersen

Problemen bij het beheersen en het schatten van kosten en doorlooptijd treft men doorgaans aan bij projecten die worden gekenmerkt door *onzekerheid* en als gevolg daarvan door grote technische, financiële en organisatorische risico's. Bekende voorbeelden hiervan zijn de aanleg van de Deltawerken, de bouw van zeeboten, vliegtuigen en tal van andere (produkt-)innovatieprojecten.

Bij de beheersing van software-ontwikkeling speelt onzekerheid, gezien het specifieke karakter van software, een zeer belangrijke rol. We hebben immers te maken met een abstract, niet tastbaar en niet zichtbaar produkt. Bij de ontwikkeling van software is het moeilijk de specificaties vooraf volledig in kaart te brengen. Het gaat er bij de ontwikkeling van software om de lijst van eisen en wensen tijdens het ontwikkeltraject duidelijk en volledig te maken. Een ander kenmerk dat hierbij nauw aansluit is de instabiliteit van specificaties. Als de werkelijkheid, waarvoor de software is gemaakt, verandert, zal de software aangepast moeten worden. Dergelijke veranderingen en de hiermee gepaard gaande kosten zijn vooraf moeilijk of zelfs niet in te schatten.

De problemen die men met behulp van automatisering tracht op te lossen zijn doorgaans complex. De programma's die hiervoor moeten worden geschreven zijn vaak groot in omvang. Dergelijk grote programma's worden door teams van soms honderden ontwikkelaars gemaakt. Het testen en onderhouden van software laat ook duidelijke verschillen zien met fysieke produkten. Het aantal toestanden in software is vaak dusdanig groot dat het tes-

INFORMATIE JAARGANG 32 NR. 2
PAG. 105 T/M 232 1990

ten van alle toestanden onbegonnen werk is. Bij het herstellen van fouten heeft software de vervelende eigenschap dat de verbetering van de ene fout de introductie van andere fouten tot gevolg heeft.

In dit artikel wordt nader ingegaan op het aspect 'onzekerheid' bij software-ontwikkeling en wordt aangegeven wat de gevolgen van onzekerheid zijn voor het beheersen en begroten.
Bij onzekerheid maken wij een onderscheid in onzekerheid wat betreft:
- het *produkt*;
- het *produktieproces* en
- de *produktiemiddelen*.

2.1 Onzekerheid produkt

De produktonzekerheid wordt in belangrijke mate bepaald door de *duidelijkheid* en de *stabiliteit* van de informatiebehoeften. Duidelijkheid wil zeggen dat men in staat is precies te omschrijven aan welke eisen de te ontwikkelen software moet voldoen. Stabiliteit wil zeggen dat de informatiebehoeften, in de tijd gezien, niet of nauwelijks veranderen.

Davis (1982) geeft aan dat de onzekerheid in het proces van informatiebehoeftebepaling met name wordt bepaald door karakteristieken van de omgeving waarvoor de software wordt ontwikkeld. Voorbeelden van factoren die de onzekerheid vergroten zijn onder andere: ontbreken van duidelijke processen in de omgeving, weinig routinewerkzaamheden en weinig stabiliteit in de structuur van het desbetreffende deel van de organisatie.
Naast onzekerheid over het bestaan van duidelijke systeemspecificaties en de stabiliteit van deze specificaties, noemt Davis ook de mate waarin gebruikers informatiebehoeften kunnen specificeren en analisten informatiebehoeften kunnen achterhalen. Zo heeft de gebruiker de neiging zich primair te laten leiden door eigen voorkeur en belangen in plaats van de belangen van de organisatie. Het gevaar bestaat dat hij daardoor relevante eisen over het hoofd ziet. De gebruiker is verder te beperkt om exact te zijn bij het formuleren van zijn of haar behoefte aan gegevens. Vaak komt het voor dat eisen in vage bewoordingen zijn geformuleerd. Bij de vertaling van de eisen naar ontwerp en code ontstaan dan interpretatieproblemen. Tenslotte noemt Davis de eigenschap dat de gebruiker een groter belang hecht aan zaken die onlangs hebben plaatsgevonden. De structurele informatiebehoefte blijft daardoor enigszins verborgen.

Van Vliet (1987) signaleert eveneens dat het specificeren van informatiebehoeften moeilijk is. Het is veelal niet voldoende de bestaande situatie als enige leidraad te gebruiken. Een belangrijke reden waarom een organisatie over wenst te gaan tot automatisering is immers in heel wat gevallen onvrede met de bestaande situatie. Een en ander betekent dat gebruiker en software-ontwikkelaar een beeld moeten krijgen van de gewenste toekomstige situatie. Dit is moeilijk, met name in die situaties, waarin de gebruiker weinig ervaring heeft met automatisering, de ontwikkelaar niet bekend is met het toepassingsgebied ('niet-materiekundig') en de innovatiegraad van de te ontwikkelen software hoog is.

Ongeacht of de informatiebehoeften wel of niet goed in kaart zijn te brengen, geldt dat bij veel applicaties de behoeften van de gebruikers evolueren. Dit betekent dat de software die wordt ontwikkeld, afgestemd wordt op een zich steeds wijzigend doel. Vaak wordt dit verschijnsel bij de ontwikkeling van software niet voldoende onderkend. Tijdens de ontwikkeling worden dan pogingen ondernomen de software te laten voldoen aan nieuwe eisen, waarop het oorspronkelijke ontwerp niet was afgestemd. Een gevolg hiervan kan zijn dat kosten en doorlooptijd fors overschreden worden. Een ander uiterste is dat ontwikkelaars de eisen hebben bevroren en er niet van doordrongen zijn dat wijzigingen onvermijdelijk zijn.
Gevolg daarvan kan zijn dat de software bij oplevering niet voldoet aan de verwachtingen van de gebruikers.

2.2 Onzekerheid produktieproces

De onzekerheid wat betreft het produktieproces wordt voor een belangrijk deel bepaald door de *mogelijkheid van bijsturing* (besturingsvariëteit) en het *inzicht in de effecten* van mogelijke stuuracties.

Bij de ontwikkeling van software zijn de mogelijkheden voor bijsturing situatie-afhankelijk. Ik wil twee extreme situaties schetsen. De eerste situatie zou men kunnen omschrijven als: de begroting ligt vast, maximaliseer het resultaat. In de tweede situatie liggen de zaken juist andersom. Het resultaat is precies bekend en men kan streven naar optimaal gebruik van middelen. Welke situatie van kracht is, wordt in belangrijke mate bepaald door de mate waarin de specificaties bekend zijn. Zijn de specificaties vaag en onvolledig, dan is er sprake van de eerste situatie. Men beschikt over mogelijkheden om bij te sturen in de functionaliteit en kwaliteit van de software. Naarmate minder aan de kwaliteit van het eindresultaat getornd mag worden en budget en tijdstip van levering minder vastliggen, treedt een verschuiving op van de eerste naar de tweede situatie. De bijstuurmogelijkheden zijn nu anders van aard, namelijk: optimaliseer naar middeleninzet en doorlooptijd.
Effectief bijsturen betekent verder dat onder andere bekend moet zijn wat het effect is van de inzet van extra personeel, van werkvoorbereidingsacties, van een ver-

hoging of verlaging van het budget, het gebruik van vierde-generatie hulpmiddelen, enz. Een besturingsmodel impliceert naast inzicht in de effecten van allerlei stuuracties, inzicht in de stand van zaken van een project. Waarom zijn budgetten en levertijden overschreden, waarom voldoet de software niet volledig aan de gestelde eisen, is er voldoende gekwalificeerd personeel ingezet, is de invloed van produktiviteitsfactoren juist ingeschat? Vaak zijn organisaties niet in staat op dergelijke vragen een antwoord te geven.

2.3 Onzekerheid produktiemiddelen

Bij de onzekerheid van de produktiemiddelen speelt de *beschikbaarheid* van de produktiemiddelen een belangrijke rol. Personeel is bij software-ontwikkeling verreweg het belangrijkste produktiemiddel. Het dynamisch karakter van het aanbod en de beschikbaarheid van automatiseringspersoneel is kenmerkend voor software-ontwikkeling. Door de snelle ontwikkelingen is het personeel genoodzaakt een groot deel van zijn werktijd te besteden aan opleiding en bijscholing. De beschikbaarheid van ontwikkelpersoneel is daardoor relatief laag. Omdat de economische waarde van automatiseringspersoneel hoog is, kan een organisatie zich veelal niet permitteren een grote overcapaciteit aan te houden.

3 Beheerssituaties

In paragraaf 2 is gesignaleerd dat de mate van *produktonzekerheid* wordt bepaald door:

1 het bestaan van duidelijke en volledige informatiebehoeften;
2 de stabiliteit van de informatiebehoeften.

De mate van *procesonzekerheid* wordt bepaald door:

3 de mogelijkheden om het proces bij te sturen;
4 het meten en kennen van effecten van bijsturen.

De mate van *middelenonzekerheid* wordt bepaald door:

5 de beschikbaarheid van personeel.

De beheersbaarheid van software-ontwikkeling, dan wel de nauwkeurigheid van een begroting hangt af van de mate van produkt-, proces- en middelenonzekerheid. Naarmate de onzekerheid toeneemt, zal de beheersbaarheid afnemen.

Door de vijf aspecten van produkt-, proces- en middelenonzekerheid te combineren, ontstaat een groot aantal combinaties. De twee meest extreme combinaties zijn die, welke voldoen aan de volgende voorwaarden:

- *Combinatie 1:* de informatiebehoeften zijn stabiel, precies te omschrijven, men 'kent' het proces, er zijn voldoende mogelijkheden tot bijsturing en er zijn geen problemen met de beschikbaarheid van het personeel. Voor beheersen en begroten van software-ontwikkeling is dit de meest gemakkelijke situatie.
- *Combinatie 2:* de informatiebehoeften zijn niet precies te omschrijven en zijn bovendien sterk aan verandering onderhevig. Procesonzekerheid is hoog, met andere woorden: er zijn relatief weinig mogelijkheden om het proces bij te sturen en de effecten van stuurmaatregelen te meten. Tenslotte zijn er problemen met de beschikbaarheid van personeel: dat wil zeggen grote middelenonzekerheid.

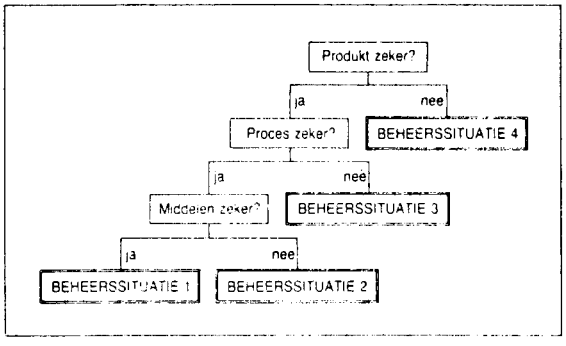
De overige mogelijke combinaties vormen een continuüm met deze twee combinaties als uitersten. Om een ordening aan te brengen in dit groot aantal combinaties heb ik de twee aspecten onduidelijkheid en instabiliteit van de informatiebehoeften samengevoegd tot één variabele *produktonzekerheid* en de twee aspecten mogelijkheid voor bijsturing en inzicht in de effecten van bijsturing tot de variabele *procesonzekerheid*. Het aspect beschikbaarheid personeel wordt verder aangeduid als de variabele *middelenonzekerheid*. Gemakshalve ga ik ervan uit dat zo'n variabele twee waarden kan hebben: de onzekerheid is hoog of de onzekerheid is laag.
Door de mogelijke waarden van de drie variabelen te combineren ontstaan acht mogelijkheden. In tabel 1 worden deze mogelijkheden genoemd.

Niet alle mogelijkheden leiden tot geldige of bestaande beheerssituaties. Mogelijkheden 5 en 6 vervallen, omdat ik uitga van de aanname dat, als er veel onzekerheid bestaat over de te ontwikkelen software, er geen sprake kan zijn van lage onzekerheid omtrent het uit te voeren proces of de benodigde middelen. Als zodanig zijn mogelijkheid 5 en 6 niet relevant om verder te analyseren. Mogelijkheid 4 en 7 vervallen om soortgelijke redenen. Als niet bekend is op welke wijze het ontwikkelproces uitgevoerd moet worden, is het niet mogelijk dat er ze-

variabelen	*mogelijkheden*							
	1	2	3	4	5	6	7	8
produktonzekerheid	laag	laag	laag	laag	hoog	hoog	hoog	hoog
procesonzekerheid	laag	laag	hoog	hoog	laag	laag	hoog	hoog
middelenonzekerheid	laag	hoog	hoog	laag	laag	hoog	laag	hoog

Tabel 1: Acht mogelijke beheerssituaties

kerheid bestaat over de middelen die bij dit proces ingezet moeten worden. Door uit te gaan van de hiërarchie 'produkt - proces - middelen' worden de acht mogelijkheden teruggebracht tot vier. Deze hiërarchie is in figuur 1 weergegeven.



Figuur 1: Een hiërarchie van beheerssituaties

De vier combinaties die aldus worden geselecteerd zullen verder worden aangeduid als de vier ideaaltypische beheerssituaties. De vier beheerssituaties die aldus ontstaan zijn:

- beheerssituatie 1: produkt, proces en middelen zeker;
- beheerssituatie 2: produkt en proces zeker, middelen onzeker;
- beheerssituatie 3: produkt zeker, proces en middelen onzeker;
- beheerssituatie 4: produkt, proces en middelen onzeker.

De vier ideaaltypische beheerssituaties zullen in paragraaf 4 nader worden toegelicht. Per beheerssituatie zal worden aangegeven welke wijze van beheersen en begroten hierbij het beste aansluit.

4 Een typologie van beheerssituaties

In paragraaf 3 zijn vier ideaaltypische beheerssituaties onderscheiden. Het criterium voor dit onderscheid is de mate van onzekerheid over het produkt, het proces en de middelen. Gaande van situatie 1 naar situatie 4 neemt de onzekerheid geleidelijk toe. Als gevolg van deze stijgende onzekerheid zal de beheersbaarheid van de softwareontwikkeling en de nauwkeurigheid van de begroting geleidelijk afnemen. In figuur 2 is de samenhang tussen onzekerheid en beheersbaarheid/nauwkeurigheid van de begroting in beeld gebracht.

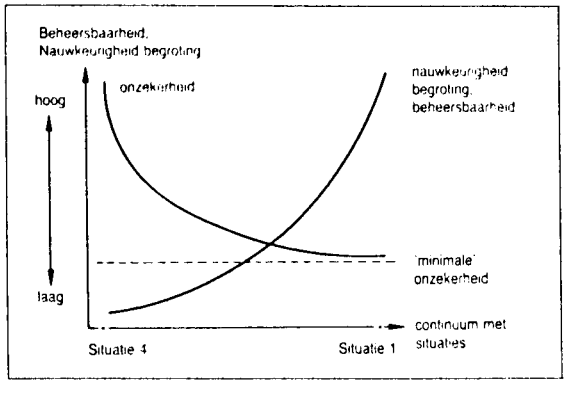
Wil men in staat zijn het proces van software-ontwikkeling effectief te beheersen, dan zal men in ieder geval inzicht dienen te hebben in de mate van onzekerheid met andere woorden dienen te weten welke beheerssituatie van kracht is. Dit inzicht alleen is echter niet voldoende om softwareprojecten beter te beheersen. Hiervoor is ook nodig dat men weet op welke wijze software-ontwikkeling in de verschillende ideaaltypische situaties moet worden aangestuurd.

In deze paragraaf wil ik de kenmerken van de vier beheerssituaties beschrijven en per beheerssituatie aangeven welke wijze van beheersen en begroten het meest effectief is. Om dit te realiseren zal ik vanuit vijf verschillende invalshoeken de vier beheerssituaties toelichten. Deze invalshoeken zijn:

- de aard van het beheersprobleem;
- het primaire doel van de beheersing;
- de wijze van coördinatie en de vorm van leiderschap;
- de te hanteren ontwikkelstrategie;
- de betekenis van een begroting en de methode van begroten.

Alvorens de vier beheerssituaties nader uit te werken, zal ik eerst deze vijf invalshoeken toelichten. Hierbij zal duidelijk worden dat per situatie een andere invulling moet worden gegeven aan wijze van coördineren, functie van een begroting enzovoorts, en dat aard van het probleem en doel van beheersing per situatie volkomen verschillend is.

Het eerste uitgangspunt heeft betrekking op de *aard van het beheersprobleem*. Zo zal men er rekening mee dienen te houden dat de beheersingswijze afgestemd is op de aard van het beheersingsprobleem. Heeft men bijvoorbeeld geen duidelijkheid over het eindresultaat, problemen met het kiezen van de weg om het eindresultaat te bereiken en met de beschikbaarheid van de benodigde middelen, dan zal een dergelijke project op een andere manier aangestuurd moeten worden dan wanneer derge-



Figuur 2: De relatie tussen enerzijds produkt-, proces- en middelenzekerheid en anderzijds de nauwkeurigheid van de begroting en de beheersbaarheid. De 'minimale' onzekerheid heeft betrekking op kenmerken die inherent zijn aan software c.q. softwareontwikkeling

lijke problemen niet of nauwelijks aanwezig zijn. In het eerste geval ligt het accent op de verkenning en formulering van een probleemsituatie, terwijl in het tweede geval de aandacht primair gericht is op de beheersing van de uitvoering van het project en op de bewaking van de produktkwaliteit.

De wijze van beheersing en begroting wordt ook bepaald door het *primaire doel* dat men wenst te realiseren bij de ontwikkeling van de software. Als het accent bijvoorbeeld op de leversnelheid ligt, dan zal dat om een andere aanpak van beheersen vragen dan in situaties waarin leversnelheid geen kritische factor is en het accent ligt op het maximaliseren van de kwaliteit van het eindresultaat.

Een derde invalshoek die een belangrijke rol speelt bij beheersing is *leidinggeven en coördinatie*. De vorm van leiderschap en de wijze van coördinatie moet zijn afgestemd op de betreffende situatie. Zo zal een team van professionele analisten, dat helderheid tracht te krijgen in de probleemsituatie en oplossingsalternatieven, op een volkomen andere wijze moeten worden aangestuurd dan ontwikkelaars die aan de hand van duidelijke specificaties en werkopdrachten een klus moeten klaren. In het laatste geval is een hiërarchisch leiderschap doorgaans de meest effectieve leiderschapsvorm. Het uit te voeren werk is veelal goed te coördineren door middel van standaardisatie. Bij omvangrijke en innovatieve projecten, waarbij sprake is van een grote mate van onzekerheid, zal een andere vorm van coördinatie en leiderschap het meest effectief zijn. Coördinatie kan het best worden gerealiseerd door onderlinge afstemming. Een dergelijke situatie vraagt om een leiderschap dat mensen weet te motiveren en te stimuleren in specialistisch en innovatief werk.

Een vierde aspect dat bepalend is voor de wijze van beheersen en begroten is de manier of *strategie* die men kiest voor het ontwikkelen van software. Is er sprake van stabiele en duidelijke informatiebehoeften, dan is het goed mogelijk het project op te delen in strikt achtereenvolgende fasen en duidelijke meetpunten te identificeren. De voortgang wordt afgemeten aan het bereiken van deze meetpunten. Naarmate men preciezer weet hoe het eindresultaat eruit ziet, kan men zich veroorloven het aantal meetpunten te reduceren en dientengevolge een minder dwingende vorm van projectbeheersing toe te passen (de lineaire strategie). Een totaal andere situatie treedt op als de stabiliteit van de informatiebehoeften gering is. Van een duidelijke fasering en onderscheid in activiteiten is nu minder sprake. De informatiebehoeften zijn voortdurend in beweging. Om de ontwikkeling beheersbaar te houden is het verstandig veel meetpunten in het project in te bouwen en de ontwikkeling in kleine stappen te laten verlopen (de incrementele strategie) (McCracken en Jackson, 1982).

De vijfde en laatste invalshoek die ik wil noemen, heeft betrekking op de *functie van een begroting* bij het beheersen van automatiseringsprojecten. Naarmate er meer betrouwbare informatie beschikbaar is om een schatting te maken van benodigde ontwikkeltijd en -inspanning, zal een begroting beter kunnen dienen als stuurinstrument voor het management. In een dergelijke begroting dienen duidelijke afspraken te worden gemaakt over de prestaties die men van de software verwacht en met welke inzet van mensen, middelen, tijd en geld deze prestaties gerealiseerd dienen te worden. De begroting is taakstellend en heeft de betekenis van een norm (Theeuwes, 1987). Anders ligt het als er een gebrek is aan gegevens voor het opstellen van een begroting. Een dergelijke situatie treft men met name aan bij aanvang van een project. De begroting kan nu niet dienen als instrument om het project te beheersen, maar heeft veel meer een voorwaardenscheppende betekenis. Het gaat in dit geval om de vaststelling van de haalbaarheid van een project. Een dergelijke begroting heeft slechts een indicatief karakter om onder andere het financiële risico in te schatten en dient als basis voor een go/no-go-beslissing.

Een en ander betekent dat de functie die men aan een begroting toekent in overeenstemming moet zijn met de geldende situatie. In de praktijk heb ik meer dan eens ervaren dat in onzekere situaties begrotingen ten onrechte een normstellende betekenis kregen, met alle nare gevolgen van dien. Net zo goed als de begrotingsfunctie situationeel wordt bepaald, zal dit ook gelden voor de wijze van begroten. De meest gangbare wijzen zijn de expertmethode, de analogiemethode en begroten met behulp van begrotingsmodellen. Bij begrotingsmodellen wordt de schatting van benodigde tijd, geld en middelen weergegeven als functie van een aantal variabelen. Bij begroten op basis van analogieën baseert degene die de begroting opstelt zich expliciet op gegevens van vergelijkbare oude projecten of vergelijkbare delen of modules hiervan. Bij de expertmethode tenslotte gaat men af op het advies van een of meer experts. De keuze voor een bepaalde methode is onder andere afhankelijk van het produkt dat men wil maken en het proces en de middelen die men daarvoor nodig heeft. Hoe geringer dit inzicht, met andere woorden hoe groter de onzekerheid, hoe minder men gebruik zal kunnen maken van een formele methode als een model; men zal zijn toevlucht moeten nemen tot het raadplegen van deskundigen.

4.1 Beheerssituatie 1

Begroten en beheersen van software-ontwikkeling is het eenvoudigst als zich de volgende situatie voordoet:

- de informatiebehoeften waaraan de software moet voldoen zijn precies te omschrijven;
- de informatiebehoeften wijzigen in de tijd niet of nauwelijks;
- het is bekend op welke wijze de software ontwikkeld moet worden; men 'kent' het proces;
- men beschikt over voldoende besturingsvariëteit en bekend is wat de gevolgen zijn van stuuracties;
- de beschikbaarheid van produktiemiddelen is goed te reguleren.

In een dergelijke 'ideale' situatie hebben we te maken met *rationele, routinematige* beheersing.
Het accent bij de ontwikkeling ligt op de *realisatie*, met andere woorden: hoe kan men, gegeven het pakket van eisen, het eindresultaat op een *zo efficiënt* mogelijke wijze bereiken. De aandacht is niet primair gericht op de analyse van de probleemsituatie, de inventarisatie van de informatiebehoeften of het opstellen van de lijst van eisen, maar op de beheersing van de uitvoering van het project en op de bewaking van de produktkwaliteit.
In een dergelijke situatie kan coördinatie het best worden gerealiseerd via een lijnorganisatie. Mintzberg (1983) spreekt zich in deze situatie uit voor *direct supervision*. Besturingsvariëteit moet gezocht worden in de verdeling van taken, de toewijzing van taken aan mensen, de inrichting van de communicatiestructuur, enzovoorts. De stijl van leidinggeven die in deze situatie het meest effectief is, is de zogenaamde *afscheidingsbasisstijl* (Redin, 1973). Het uit te voeren werk ligt vast in regels, procedures en voorschriften, met andere woorden heeft een routinematig karakter. Voor de realisatie kan men volstaan met lager gekwalificeerd personeel. De taak van de leidinggever, in dit geval de projectmanager, ligt vooral op het toekennen van opdrachten en het controleren van de uitvoering ervan.
In een dergelijke zekere en stabiele situatie kan men bij de ontwikkeling van software gebruik maken van een *lineaire* ontwikkelstrategie; dat wil zeggen de opdeling van de ontwikkeling in strikt achtereenvolgende fasen. Door de hoge mate van zekerheid is men goed in staat het project op te splitsen in fasen, activiteiten en taken, en kan men per fase/activiteit/taak het resultaat precies specificeren en aangeven welke personen met welke vaardigheden een en ander moeten uitvoeren.
Als de produkt-, proces en middelenzekerheid groot is kan men bij het opstellen van een begroting gebruik maken van een sterk geformaliseerde werkwijze. Dit betekent dat *begrotingsmodellen* hier goed bruikbaar zijn. Een dergelijke begroting kan dienen als instrument om een softwareproject te beheersen dan wel te *bewaken*. De begroting heeft primair een *taakstellende* functie. In deze begrotingssituatie kan een *databank* met voltooide projectgegevens een geschikt hulpmiddel zijn.

4.2 Beheerssituatie 2
Deze situatie treedt op als de eisen waaraan de software moet voldoen duidelijk, volledig en stabiel zijn. Men heeft tevens een goed inzicht op welke wijze het eindresultaat bereikt moet worden. Men beschikt over voldoende besturingsvariëteit en kent de effecten van stuuracties. Er is echter onzekerheid wat betreft de produktiemiddelen.
Het kernprobleem bij de ontwikkeling van software in een dergelijke situatie spitst zich toe op de *beschikbaarheid van personeel.*
Beheersen tendeert naar het oplossen van een *capaciteitsvraagstuk*. Vragen die hierbij een rol spelen, zijn onder andere: hoe krijgt men het project bemand, waar haalt men geschikt personeel vandaan, hoe kan men met de beperkte middelen het gewenste eindresultaat leveren. Wij zullen niet nader ingaan op mogelijkheden van scholing, salariëring en loopbaanplanning.
Een mogelijke strategie die gevolgd kan worden bij beschikbaarheidproblemen van adequaat personeel is het *uitbesteden* van het werk. In de optiek van Mintzberg zal men, in geval de stabiliteit van het personeel gering is, ernaar moeten streven het proces zoveel mogelijk te standaardiseren (*standardization of work processes*). Hierdoor wordt de uitwisselbaarheid of vervangbaarheid vergroot. Richtlijnen en procedures schrijven voor hoe de verschillende taken uitgevoerd moeten worden. Het gevolg hiervan is dat volstaan kan worden met lager geschoold personeel. Dergelijke maatregelen kunnen alleen effect hebben als de produkt- en procesonzekerheid gering is en de te ontwikkelen software bovendien niet te complex is. Naarmate minder aan deze voorwaarden wordt voldaan, nemen de risico's voor het project sterk toe.
Voor de begroting kan men evenals bij beheerssituatie 1 gebruik maken van *begrotingsmodellen* en *gegevens van voltooide projecten*. Omdat er onzekerheid bestaat omtrent de produktiemiddelen, zal men bij het opstellen van een begroting behoefte hebben aan *gevoeligheidsanalyses*. Met behulp van dergelijke analyses is men in staat na te gaan wat het effect op kosten en doorlooptijd is als er wijzigingen optreden in de in te zetten middelen. Bijvoorbeeld: hoe hoog worden de kosten als in plaats van vier ontwerpers met een bepaald opleidings- en ervaringsniveau, drie ontwerpers worden ingezet van een ander niveau'.

4.3 Beheerssituatie 3
Deze situatie treedt op als de informatiebehoeften precies omschreven kunnen worden en gedurende langere tijd stabiel zijn. Het is echter niet duidelijk op welke wijze het project uitgevoerd moet worden, dus hoe het resultaat bereikt moet worden. Men heeft geen inzicht in de gevolgen van mogelijke stuuracties. Er zijn evenals in

beheerssituatie 2 problemen wat betreft de beschikbaarheid van personeel.
Er is hier sprake van een *ontwerpprobleem*. Er bestaan geen onzekerheden meer over het te ontwikkelen produkt, maar wel over het uit te voeren proces en de in te zetten middelen. Met ontwerpprobleem wordt bedoeld dat antwoord gegeven moet worden op vragen als: wordt het project wel of niet projectmatig aangepakt; welke beslispunten worden in het project ingebouwd; welke documenten moeten wanneer worden opgeleverd; hoe moeten mensen worden ingezet; hoe worden verantwoordelijkheden en bevoegdheden verdeeld enzovoorts.
Een en ander betekent dat wat de factoren betreft die bepalend zijn voor de ontwikkelkosten, duidelijkheid bestaat over de zogenaamde produkt- en gebruikersafhankelijke factoren (Heemstra, 1987). Tijdens de ontwikkeling is sturing mogelijk door invloed uit te oefenen op de middelen-, project- en personeelsafhankelijke factoren. Wat de invloed van die beïnvloedbare factoren op de kosten en doorlooptijd is, is niet bekend. Dit betekent dat men onvoldoende inzicht heeft in de gevolgen van het inzetten van extra personeel, andere hulpmiddelen, andere methoden en technieken enzovoort. Het accent bij de uitvoering ligt op de *beheersing* van het ontwikkelproces.
Het coördinatiemechanisme van Mintzberg dat het beste aansluit bij deze situatie, is het *standaardiseren van de output*. De uitvoer is immers gespecificeerd. Omdat de kwaliteit van de te ontwikkelen software 'vast ligt', is beheersing hoofdzakelijk mogelijk via sturing van het proces en de middelen. Over deze twee zaken bestaat echter onzekerheid.
Wil men de software-ontwikkeling in dergelijke zekere omstandigheden beheersbaar houden, dan zal men met twee zaken rekening moeten houden. Allereerst zal men genoodzaakt zijn overcapaciteit in te bouwen. Wat betreft het proces laat zich dat onder andere vertalen in *marges* in ontwikkeltijd en -budget. Voor middelen betekent dit bijvoorbeeld extra personeel inschakelen, parallelle ontwikkelteams inzetten enzovoorts. Omdat deze situatie zich kenmerkt door een geringe beschikbaarheid van personeel, is het aanhouden van overcapaciteit niet opportuun. Ten tweede zal men als gevolg van de onzekerheid het aantal meetpunten en de meetfrequentie groot moeten maken. In deze situatie ligt de overgang van de *lineaire ontwikkelstrategie* naar de *incrementele*. Naarmate de onzekerheid toeneemt wordt de voorkeur voor de incrementele strategie groter.
Om in een dergelijke situatie te kunnen begroten, is het van belang dat degene die de begroting opstelt, beschikt over *gegevens van afgesloten projecten*. In zo'n gegevensverzameling moet worden gezocht naar softwareprojecten die wat betreft het *wat* en *voor wie* gelijkenis vertonen met het uit te voeren project. Aan de hand van deze informatie wordt inzicht verkregen in het gedrag van de *waarmee, hoe-* en *wie*-factoren in vergelijkbare situaties (Heemstra, 1987).
Evenals in beheerssituatie 2 zal men bij het opstellen van een begroting gebruik moeten maken van *gevoeligheidsanalyses*. Omdat de onzekerheid in deze derde situatie groter is dan in beheerssituatie 2, zal de noodzaak voor dergelijke analyses ook groter zijn. Voor het projectmanagement is het veel interessanter als er aangegeven kan worden hoe gevoelig een begroting is voor bepaalde invloedsfactoren. Bijvoorbeeld: wat is het effect op de doorlooptijd van de inzet van twee extra analisten; hoe ontwikkelen zich de kosten als de projectduur met *x* dagen wordt ingekort?

4.4 Beheerssituatie 4

De moeilijkste situatie bij de beheersing van softwareontwikkeling en het opstellen van een begroting treedt op als er sprake is van een grote onzekerheid en wazigheid. Bij de ontwikkeling van software, met name in de beginfasen, treedt dit verschijnsel maar al te vaak op. Een en ander betekent dat er geen duidelijkheid bestaat over het eindresultaat, noch over de weg waarlangs dit eindresultaat bereikt moet worden en noch over de middelen die men hierbij kan inzetten. Ook heeft men geen duidelijk idee welke factoren bepalend zijn voor de kosten en doorlooptijd; men heeft geen inzicht in het waardenbereik van de verschillende factoren.
Vanwege de produktonzekerheid zal een belangrijk deel van de ontwikkelinspanning gaan zitten in een verkenning van de probleemsituatie en een analyse van de informatiebehoeften. Het uit te voeren werk is vooral *exploratief* van aard.
De voorwaarden om het ontwikkelproces te beheersen zijn minimaal. Men kan immers niet duidelijk aangeven welke software ontwikkeld moet worden. De wijze waarop en de middelen waarmee de software gerealiseerd moet worden zijn als gevolg daarvan eveneens niet duidelijk. Om het uit te voeren werk alsnog beheersbaar te maken, is het aan te bevelen de middelen vast te leggen/zeker te maken. Het doel wordt dan om met deze *gegevens middelen het resultaat te maximaliseren.*

In de optiek van Mintzberg is coördinatie in een dergelijke situatie niet te realiseren met behulp van standaardisatie. Het eindresultaat is niet precies te omschrijven en is niet stabiel: coördinatie door standaardisatie van de output is daarom niet mogelijk. De weg waarlangs het resultaat bereikt moet worden is niet bekend: coördinatie door standaardisatie van het proces vervalt. Beschikbaarheid van personeel is een probleem: coördinatie door standaardisatie van in te zetten personeel is moeilijk uitvoerbaar. Coördinatie kan in een dergelijke com-

plexe en onzekere situatie het best gerealiseerd worden door *mutual adjustment*.
Verder signaleert Mintzberg dat voor de hier beschreven situatie *adhocracy* de beste structuurconfiguratie is. Dynamiek, innovatie, complexiteit, onzekerheid en risico's vormen de belangrijkste uitgangspunten voor adhocracy. Adhocracy wil zeggen dat *experts* van diverse disciplines in projectgroepen samenwerken. Coördinatie via standaardisatie is niet of nauwelijks mogelijk, er is weinig formalisme. Bevoegdheden en verantwoordelijkheden liggen bij de experts die het werk uitvoeren. Er is sprake van een *horizontale organisatiestructuur*. Een belangrijke factor voor het slagen van een softwareproject in een dergelijke onzekere situatie wordt in belangrijke mate bepaald door *commitment* van de uitvoerenden. Het is niet mogelijk het werk op te splitsen in keurig afgebakende activiteiten en precies omschreven modules. Door de onzekerheid is een planning en begroting per definitie onnauwkeurig. De aanpak van het werk zal meer *improviserend* van aard zijn. Bij het leidinggeven zal een afscheidingsstijl (Redlin, 1973) averechts werken. Het zich houden aan een planning/begroting kan niet worden afgedwongen, maar zal gedragen moeten worden door alle betrokkenen. Het management zal de juiste voorwaarden moeten scheppen, zodat het dure, hoog gekwalificeerd personeel haar werk kan uitvoeren. De *relatiestijl* sluit het beste aan in deze situatie. Het is van belang dat men snel kan inspelen op nieuwe, plotseling optredende omstandigheden.

KENMERKEN VAN SOFTWARE/ SOFTWARE-ONTWIKKELING	produkt zeker proces zeker middelen zeker	produkt zeker proces zeker middelen onzeker	produkt zeker proces onzeker middelen onzeker	produkt onzeker proces onzeker middelen onzeker
	BEHEERSSITUATIE 1	BEHEERSSITUATIE 2	BEHEERSSITUATIE 3	BEHEERSSITUATIE 4
UITGANGS-PUNTEN				
AARD PROBLEEM	- realisatie	- allocatie	- ontwerp	- exploratie
PRIMAIR DOEL BEHEERSING	- doel gegeven, optimaliseer middelen - efficiency en doorlooptijd	- werving, opleiding, scholing personeel - efficiënt gebruik middelen	- beheersing van het proces	- middelen gegeven maximaliseer resultaat - effectiviteit - verlagen risico
COÖRDINATIE MECHANISME, LEIDINGGEVEN	- standaardisatie produkt, proces, middelen - hiërarchie, afscheidingsstijl	- standaardisatie produkt, proces	- standaardisatie proces	- mutual adjustment - commitment - relatiestijl
ONTWIKKEL-STRATEGIE	- lineair - projectmatig - rationaliteit	- uitbesteden - middelenexperts	- incrementeel - procesexperts	- incrementeel - prototyping - improviserend
BEGROTEN	- modellen - databank oude projectgegevens - taakstellend - bewakend	- modellen - databank oude projectgegevens - gevoeligheids-analyses	- databank oude projectgegevens - gevoeligheids analyses	- expertmethode delfi-methode - risico-analyse - voorwaarden-scheppend

Figuur 3: De vier beheerssituaties

Omdat de informatiebehoeften niet duidelijk zijn, is het verstandig de ontwikkeling te beginnen met *prototyping*. De grote onzekerheid maakt het noodzakelijk dat het aantal meetpunten groot is. Hoe groter die onzekerheid, hoe vaker men zal moeten controleren of men het juiste produkt op de juiste wijze ontwikkelt. De *kleine-stapmethode* heeft in een dergelijke situatie de voorkeur.
Een onzekere situatie zoals hier beschreven, treft men vaak aan bij aanvang van software-ontwikkeling. De risico's zijn groot. Beheersen en begroten zijn dan vrijwel onuitvoerbaar. Het is noodzakelijk de risico's te bepalen en te proberen deze te verlagen om beheersen en begroten beter mogelijk te maken. Voor het bepalen van risico's kan men gebruik maken van *risico-analyses*. Het projectmanagement dient tijdens het totale systeemontwikkelingstraject op de hoogte te zijn van de risico's van het betreffende project.
Bij het opstellen van een begroting heeft het weinig zin gebruik te maken van gegevens van voltooide projecten. Zoeken naar analoge projecten veronderstelt immers dat men weet wat men zoekt, of wel de specificaties van de te ontwikkelen software bekend zijn. Ook de sterk geformaliseerde aanpak met behulp van begrotingsmodellen zal om die zelfde reden weinig succes hebben. Bij het begroten zal men vooral gebruik moeten maken van de *expertmethode* en/of een *delfi-achtige* aanpak.
Omdat de onzekerheid groot is en als gevolg daarvan de onnauwkeurigheid van de begroting groot, kan een begroting in deze situatie nooit een taakstellende functie hebben. De betekenis van een begroting zal veel eerder *voorwaardenscheppend* zijn.

Ter afronding van deze paragraaf worden in figuur 3 de vier beschreven beheerssituaties weergegeven.

5 Conclusies

In dit artikel is gesignaleerd dat de wijze waarop software-ontwikkeling beheerst en begroot dient te worden, in belangrijke mate afhankelijk is van de onzekerheid over de software die ontwikkeld moet worden, van het ontwikkelproces en van de benodigde en beschikbare middelen. Voor de keuze van de juiste beheersstrategie en begrotingsaanpak is het belangrijk inzicht te hebben in deze onzekerheid. Een belangrijke stap is daarom het karakteriseren en typeren van de software-ontwikkeling in kwestie.
In dit artikel is zo'n typologie van beheerssituaties gegeven. Het verdient aanbeveling in verder onderzoek de gepresenteerde typologie te verfijnen en de praktische waarde ervan in de praktijk te toetsen.

INFORMATIE JAARGANG 32 NR. 2
PAG. 105 T/M 232 1990

Referenties

Davis, G.B. (1982), 'Strategies for information requirements determination', *IBM Systems Journal*, jaargang 21, nr. 1.

Genuchten, M. van en M. Fierst van Wijnandsbergen (1989), 'An empirical study on the control of software development', *Conferentiebijdrage IFIP 8.2*, sept. 1989.

Heemstra, F.J. (1989), *Hoe duur is programmatuur? Begroten en beheersen van software-ontwikkeling*, Kluwer Bedrijfswetenschappen, Deventer.

Heemstra, F.J., W. Siskens en H. van der Stelt (1989(, 'Kostenbeheersing bij automatiseringsprojecten: een empirisch onderzoek', *Informatie*, jrg. 31, nr. 1.

Heemstra, F.J. (1987), 'Wat bepaalt de kosten van software', *Informatie*, jaargang 29, extra editie.

Hofstede, G. (1967), *The Game of Budget Control*, Van Gorcum, Assen en Tavistock, London.

Lierop, F. van en R. Volkers (1989), *Beheersen van software-ontwikkeling: een kwestie van meten*, afstudeerverslag Faculteit Bedrijfskunde TUE.

McCracken, D.D. en M.A. Jackson (1982), 'Life cycle concepts considered harmful', *Software Engineering Notes*, ACM, april 1982.

Mintzberg, H. (1983), *Structures in fives: designing effective organizations*, Prentice-Hall Inc., Englewood Cliffs, New Jersey.

Mintzberg, H. (1983), *Structures in fives: designing effective organizations*, Prentice-Hall Inc., Englewood Cliffs, New Jersey.

Phan, D., D. Vogel en J. Nunamaker (1989), 'The search for perfect project management', *Computerworld*, sept. 1989.

Reddin, W.J. (1973), *Managerseffectiviteit*, Samsom/Nive.

Thambain, H.J. en D.L. Wilemon (1986), 'Criteria for controlling projects according to plan', *Project Management Journal*, juni 1986.

Theeuwes, J.A.M. (1987), 'Budgettering: stuurinstrument of administratieve rompslomp?, Bedrijfskunde nr. 1.

Vliet, van J.C. (1987). 'Wat kost dat nou, zo'n programma' *Informatie*, jaargang 29, extra editie.

Vonk, R. (1987), *Prototyping van informatiesystemen*, Academic Service, Schoonhoven.

Dr. ir. Fred J. Heemstra is als Universitair Hoofddocent verbonden aan de Technische Universiteit Eindhoven, Faculteit Bedrijfskunde, Vakgroep Bestuurlijke Informatiesytemen en Automatisering.