Heeft iemand de software al gezien? : inzicht in het uitlopen van softwareprojecten

Heeft iemand de software al gezien?

Inzicht in het uitlopen van softwareprojecten

F. van Lierop, R. Volkers, M. van Genuchten en F. Heemstra

Het doel van dit artikel is tweeledig. Allereerst willen we laten zien dat het mogelijk is op een eenvoudige manier gegevens over het uitlopen van softwareontwikkelprojecten en over oorzaken van dit uitlopen te verzamelen. Ten tweede willen we duidelijk maken hoe belangrijk het is om, bij het analyseren van de verschillen tussen plan en werkelijkheid en bij het aanbevelen van oplossingen, de projectmedewerkers in kwestie te betrekken. Het artikel beschrijft een empirisch onderzoek zoals dat binnen Philips is uitgevoerd.

1 Inleiding

In de titel van dit artikel wordt gewezen op twee belangrijke kenmerken van software en software-ontwikkeling. We hebben namelijk te maken met een bijzonder produkt. Software is immers abstract, niet tastbaar en niet zichtbaar. Brooks (1987) noemt het niet-tastbare van software als één van de redenen waarom het ontwikkelen van software zo moeilijk is. Het tweede kenmerk waar de titel op duidt, heeft te maken met de beheersing van software-ontwikkeling. Software is veelal niet op de geplande datum gereed. In het juninummer van het blad *Electronics* wordt onder de welsprekende titel '*Product delays beset the industry*' een aantal recente voorbeelden genoemd van bekende softwareprodukten die te kampen hebben gehad met overschrijdingen van geplande levertijd. In tabel 1 worden deze voorbeelden gegeven.

BEDRIJF	PRODUKT	GEPLANDE RELEASE DATUM	WERKELIJKE RELEASE DATUM
Ashton-Tate	dbase IV	7/88	6/89
Lotus	1-2-3, rel. 3.0	6/88	6/89
	1-2-3, rel. 2.2	6/88	9/89
Microsoft	Word 4.0 MAC	10/88	4/89
	Word 5.0 PC	1/89	5/89
	SQL server	12/88	4/89

Tabel 1: Overschrijding van geplande levertijd bij de ontwikkeling van een aantal bekende softwareprodukten (*Electronics*, juni 1989)

Via de media bereiken ons geregeld berichten dat het lastig is om voor automatiseringsprojecten betrouwbare schattingen te maken van de benodigde hoeveelheid tijd en inspanning. De werkelijkheid blijkt meer dan eens van het plan af te wijken. Het is vaak niet duidelijk wat de oorzaken zijn van dergelijke afwijkingen. Inzicht in het *waarom* is echter een voorwaarde voor de bepaling van maatregelen om de planning van toekomstige projecten te verbeteren.

Het doel van dit artikel is dit inzicht te vergroten. Om dit te realiseren wordt een empirisch onderzoek beschreven waarin verschillen tussen plan en werkelijkheid in kaart zijn gebracht en de oorzaken van deze verschillen zijn weergegeven. Wij conformeren ons aan de literatuur (Wijnen, Storm en Renes (1987), Gilb (1988)) over projectbeheersing, waarin wordt gesteld, dat een project beheerst verloopt als het resulterende produkt voldoet aan de gestelde kwaliteitseisen en op de geplande tijd tegen de geplande kosten wordt opgeleverd. Het onderzoek dat in dit artikel wordt beschreven meet in een project per activiteit verschillen tussen plan en werkelijkheid en gaat uit van de vooronderstelling dat zo'n activiteit pas is voltooid als het resultaat van de activiteit voldoet aan de specificaties; met andere woorden: als aan de gestelde kwaliteitseisen is voldaan.

Het inzicht dat door het verzamelen en analyseren van de projectgegevens wordt verkregen wordt op twee manieren aangewend:

- ten eerste wordt het inzicht gebruikt om gerichte verbeteringsmaatregelen te bepalen. Dit wordt beschreven in paragraaf 2;
- ten tweede wordt het inzicht in de oorzaken van verschillen tussen plan en werkelijkheid gebruikt om een individueel project te bewaken. Hierbij wordt gebruik gemaakt van de zogenaamde S-curve (Harrison (1977)). Dit is een bekende techniek om het verloop van een project in kaart te brengen. In paragraaf 3 wordt hierop nader ingegaan.

Dit artikel wil op twee manieren een bijdrage leveren aan een betere beheersing van software-ontwikkeling. In de eerste plaats door de presentatie van een aanpak om empirische gegevens over software-ontwikkeling te verzamelen. In de tweede plaats door te laten zien dat het verzamelen van gegevens met een eenvoudig meetinstrument binnen korte tijd tot resultaten leidt. Zowel in de literatuur als in de praktijk wordt de noodzaak van het verzamelen van gegevens over software-ontwikkeling onderkend. Desondanks heeft dit in veel gevallen nog niet geleid tot het daadwerkelijk verzamelen van gegevens. Wellicht dat dit voorbeeld anderen aanspoort in hun eigen omgeving op een vergelijkbare, eenvoudige manier gegevens over software-ontwikkeling te gaan verzamelen.

Wil meten en verbeteren aan software-ontwikkeling succesvol zijn, dan is het belangrijk dat voldaan wordt aan de voorwaarde van de zogenaamde '*gesloten-lus-automatisering*' (Bemelmans (1989)). Dit betekent dat de meetcijfers vooral niet moeten bureaucratiseren maar de uitvoerenden moeten motiveren en de betrokkenheid bij een project opvoeren. Bij de opzet en uitvoering van dit onderzoek heeft het principe van de gesloten lus automatisering centraal gestaan.

Het beschreven empirisch onderzoek is in 1989 uitgevoerd in een samenwerkingsverband van de Technische Universiteit Eindhoven, vakgroep Bestuurlijke Informatiesyste-

INFORMATIE JAARGANG 33 NR. 3
PAG. 141 T/M 220 1991

men en Automatisering en Philips, afdeling Centre For Technology, Industrial Software Production Centre.

2 Analyse van software-ontwikkeling

In deze paragraaf wordt aangegeven hoe het meten aan software-ontwikkeling kan leiden tot maatregelen om de beheersing van software-ontwikkeling te verbeteren. De meting is eenvoudig van aard en beperkt zich tot het bepalen van verschillen tussen plan en werkelijkheid en tot het signaleren van de oorzaken van deze verschillen. Verbeteringsmaatregelen voor de beheersing dienen te leiden tot gerichte acties om de geconstateerde oorzaken aan te pakken. Deze acties kunnen van structurele aard zijn als blijkt dat bij de uitvoering van projecten steeds weer dezelfde problemen optreden. Acties kunnen echter ook incidenteel van aard zijn en beperkt blijven tot één project. Om gericht aan een project te kunnen sturen, dient men te beschikken over besturingsinformatie. Voor het verzamelen van dit soort informatie hebben wij een meetinstrument ontwikkeld. Met behulp van dit meetinstrument wordt de informatie op een laag aggregatieniveau binnen het project vastgelegd. Een laag aggregatieniveau betekent in dit geval dat de metingen op activiteitniveau worden verricht. De reden hiervoor ligt voor de hand: om snel op afwijkingen tussen plan en werkelijkheid te kunnen reageren, moet ons inziens op een gedetailleerd niveau worden gemeten. Een project loopt namelijk vaker uit door vele kleine problemen dan door één groot probleem. In de woorden van Brooks (1975): *'How does a project get one year late? One day at a time !'.*

oorzaaksoort	*omschrijving / voorbeeld*
capaciteitgebonden	benodigde capaciteit is te laat/te vroeg aanwezig (voorbeeld: beschikbaarheid van projectmedewerkers)
persoonsgebonden	persoonsgebonden oorzaken hebben met name betrekking op ervaring van de projectmedewerkers
inputgebonden	er wordt niet voldaan aan de vereiste voorwaarden om de activiteit te starten (voorbeeld: de user-requirements zijn te laat)
produktgebonden	oorzaken zijn terug te voeren tot produkt (voorbeeld: specificaties technisch niet te realiseren)
organisatiegebonden	de oorzaken hebben betrekking op de wijze waarop het project is georganiseerd (voorbeeld: veel werkonderbrekingen, hoog verloop)
middelengebonden	er zijn problemen met de benodigde apparatuur en tools (voorbeeld: te geringe beschikbaarheid ontwikkelapparatuur)
overige	

Tabel 2: Oorzaaksoorten

Een centrale vraag is: over welke besturingsinformatie moet men beschikken? Met andere woorden: welke gegevens over activiteiten moet men verzamelen?
Zoals in de inleiding is aangegeven, richten wij ons op de beheersaspecten *tijd* en *kosten*. Van het beheersaspect kosten beschouwen we alleen de *menskosten*. Omdat mensuren en menskosten in hoge mate uitwisselbaar zijn, kan voor het volgen van de voortgang van een project worden volstaan met het meten van:

- geplande en werkelijke doorlooptijd en mensuren per activiteit;
- geplande en werkelijke begindatum en einddatum per activiteit;
- eventuele verschillen tussen plan en werkelijkheid.

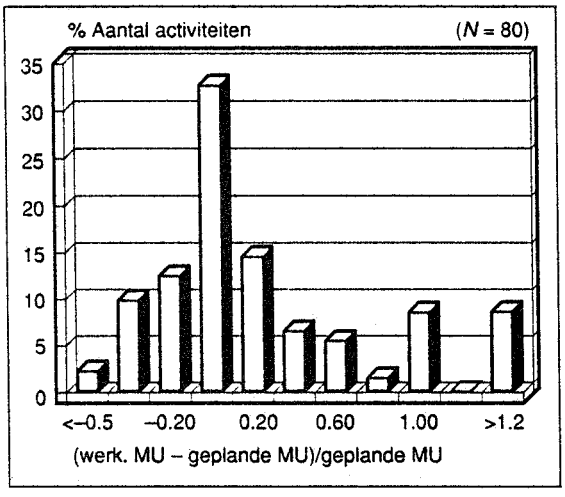
Per activiteit moeten in principe vier *verschillen tussen plan en werkelijkheid* worden verklaard: die van de mensuren, de begindatum, de einddatum en de doorlooptijd. Verschillen wat betreft begindatum en doorlooptijd van een activiteit verklaren echter al het eventueel optredende verschil in einddatum van een activiteit.
Om het aangeven van oorzaken van verschillen tussen plan en realisatie te vergemakkelijken, is een *indeling naar oorzaaksoorten* gemaakt. Deze indeling is gebaseerd op een clustering van factoren die de ontwikkelkosten van software bepalen (Heemstra (1989)). Per oorzaaksoort is een aantal oorzaken onderscheiden die relevant zijn voor de afdeling waar het meetmodel wordt toegepast. De oorzaaksoorten worden in tabel 2 vermeld. De volledige lijst met oorzaken staat vermeld in het kader aan het eind van dit artikel.

Tijdens de uitvoering van het onderzoek bij de betreffende Philipsafdeling is gemeten aan 80 activiteiten afkomstig uit 5 verschillende projecten.
De meetgegevens die betrekking hebben op plan en werkelijkheid waren betrekkelijk eenvoudig te achterhalen. Het gaat hier immers om gegevens die voor projectbeheersing elementair zijn. Zonder dit soort gegevens is elke vorm van beheersing onmogelijk. Dit ligt anders bij het verzamelen van gegevens over oorzaken van verschillen tussen plan en werkelijkheid. Dit soort gegevens is geïnventariseerd in persoonlijke gesprekken met projectleiders en medewerkers. Een belangrijke afbakening in het onderzoek is, dat alle meetgegevens betrekking hebben op activiteiten uit de implementatiefase.

De resultaten van de metingen zullen in de rest van deze paragraaf worden gepresenteerd en geïnterpreteerd. Eerst volgen de resultaten van de metingen van mensuren, daarna die van doorlooptijd. Tenslotte volgt een interpretatie.

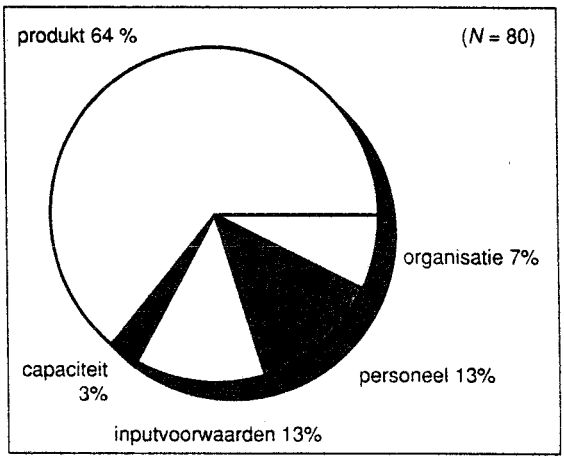
Mensuren

Voor de mensuren geldt dat het geplande aantal uren voor de 80 activiteiten met gemiddeld 20% wordt overschreden. Voor de 80 activiteiten waren in totaal 3203 uren gepland en na uitvoering bleken 3838 uren te zijn gebruikt. Voor een verdere analyse van de afwijkingen tussen plan en werkelijkheid is het belangrijk de omvang van de uitloop te relateren aan de omvang van de activiteit in mensuren. Dit betekent dat bijvoorbeeld een overschrijding van 4 uren voor een activiteit met een geplande inspanning van 8 uren niet hetzelfde gewicht heeft als eenzelfde overschrijding van 4 uren voor een activiteit met een geplande inspanning van 80 uren. Vandaar dat in figuur 1 de verschillen tussen plan en werkelijkheid worden gerelateerd aan de omvang van de activiteiten. Zo valt uit deze figuur bijvoorbeeld af te lezen dat er 14 activiteiten zijn die 20% meer mensuren hebben gekost dan gepland.



Figuur 1: Frequentieverdeling van de relatieve afwijkingen tussen gepland en werkelijk aantal mensuren; MU staat voor mensuren, de klassebreedte is 0.2; de gemiddelden van de klassen worden in de figuur weergegeven

Uit figuur 1 is af te lezen dat voor de realisatie van 21 activiteiten minder mensuren nodig waren dan gepland (dit geldt voor 25% van het totaal aantal activiteiten). 24 activiteiten zijn precies volgens plan verlopen (30%) en 35 activiteiten hebben te kampen gehad met een overschrijding (45%). Het is opvallend dat, naarmate de mate van de overschrijdingen toeneemt, het aantal activiteiten dat hierbij hoort in eerste instantie geleidelijk afneemt en later weer toeneemt. Zoals verder zal blijken geldt ditzelfde beeld ook voor de afwijkingen wat betreft doorlooptijd.



Figuur 2: Genoemde oorzaaksoorten m.b.t. overschrijding in mensuren

We hebben onderzocht of het hier altijd dezelfde soort activiteiten betrof of dat er een specifieke oorzaaksoort voor deze zogenaamde 'high end' overschrijdingen was aan te geven. Dit bleek niet het geval te zijn. Een afdoende verklaring voor dit verschijnsel kan niet worden gegeven.

In figuur 2 wordt in percentages aangegeven hoe vaak de verschillende oorzaaksoorten als reden voor afwijking tussen plan en werkelijkheid is genoemd. De meest genoemde oorzaaksoort, met 64%, heeft betrekking op produktgebonden oorzaken. De volgende oorzaken werden vaak genoemd:

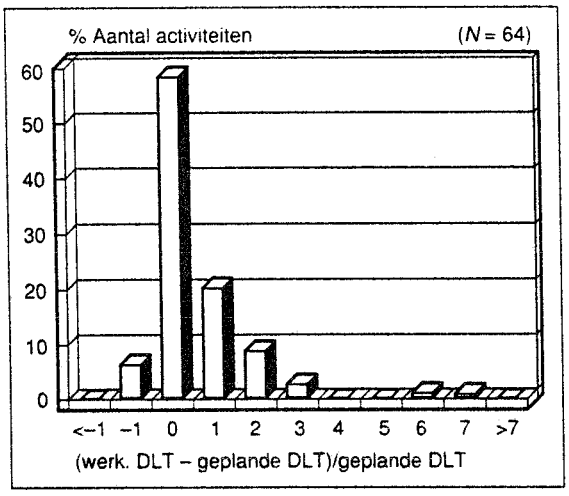
- onderschatting van de hoeveelheid werk,
- onderschatting van de complexiteit van de toepassing,
- specificaties die technisch niet realiseerbaar bleken te zijn.

Doorlooptijd

Voordat we ingaan op de meetgegevens die betrekking hebben op de doorlooptijd, willen we naar voren brengen dat in de betreffende afdeling het halen van het geplande tijdstip van levering vaak belangrijker is dan het realiseren van de software binnen het geplande budget. Als de software te laat gereed is, stagneert de ontwikkeling van het produkt waar de software een onderdeel van is.

De doorlooptijd van de activiteiten (in dagen) blijkt gemiddeld 28% langer te zijn dan gepland. Voor alle activiteiten tesamen waren 406 dagen doorlooptijd gepland, terwijl de totale doorlooptijd in werkelijkheid 526 dagen is geweest.

Net als bij de mensuren is voor de doorlooptijd de omvang van de uitloop gerelateerd aan de omvang van de activiteit. In figuur 3 worden de frequenties van de relatieve afwijkingen tussen plan en werkelijkheid weergegeven.



Figuur 3: Frequentieverdeling van de relatieve afwijkingen tussen geplande en werkelijke doorlooptijd; DLT staat voor doorlooptijd; de klassebreedte is 1; de gemiddelden van de klassen worden in de figuur weergegeven

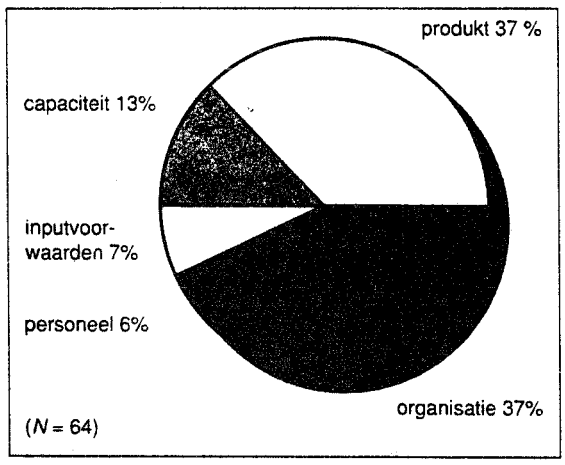
Het aantal activiteiten met een kortere doorlooptijd dan gepland, is 4 (dit is 6% van het totaal aantal activiteiten). Het aantal dat precies volgens plan is verlopen, is 37 (58%) en het aantal dat langer duurder dan gepland, is 23 (36%). De oorzaaksoorten die werden genoemd, worden weergegeven in figuur 4.

De produktgebonden oorzaken die bij 'mensuren' reeds ter sprake kwamen, werden ook voor de doorlooptijdverschillen vaak genoemd (37%). Een tweede oorzaaksoort die bij de doorlooptijd herhaaldelijk werd genoemd wordt gevormd door organisatiegebonden oorzaken (37%). Voorbeelden van deze oorzaaksoort zijn: meer werkonderbrekingen dan verwacht en het parallel lopen van activiteiten. De verdelingen van genoemde oorzaaksoorten bij 'mensuren' en 'doorlooptijden' verschillen duidelijk. De belangrijkste verschillen zijn:

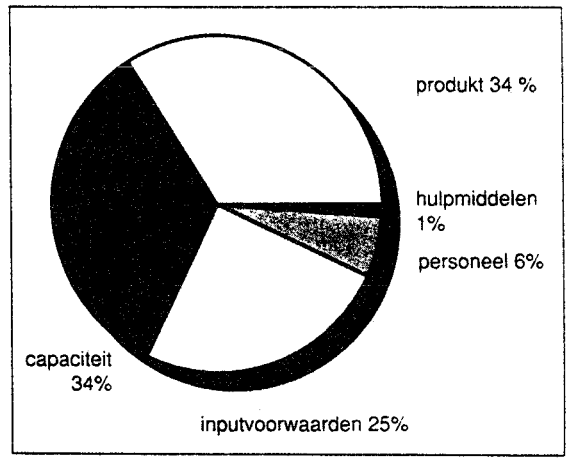
- als oorzaaksoort voor doorlooptijdverschillen worden vooral produkt- en organisatiegebonden oorzaken genoemd (beide in 37% van de gevallen);
- bij mensurenverschillen wordt in 65% van de gevallen een produktgebonden oorzaak genoemd, terwijl organisatiegebonden oorzaken slechts in 7% van de gevallen worden genoemd.

Tenslotte gaan we nader in op een analyse en interpretatie van de meetresultaten. Wij zijn ervan overtuigd dat dit soort gegevens over projecten c.q. activiteiten consequent moet worden verzameld en vastgelegd. Zij vormen een belangrijke referentie voor het begroten en beheersen van toekomstige projecten.
Verder willen wij erop wijzen dat het belangrijk is om gegevens *per* afdeling te verzamelen. Het is zeer wel mogelijk dat voor verschillende afdelingen die te kampen hebben met aanzienlijke overschrijdingen verschillende verbeteringsmaatregelen gekozen moeten worden. De oorzaak voor overschrijdingen verschilt immers doorgaans van afdeling tot afdeling. Verbeteringsmaatregelen dienen derhalve afdelingsgebonden te zijn.
Ter illustratie geven we de oorzaken van verschillen tussen plan en werkelijkheid voor de mensuren zoals die geconstateerd zijn bij een andere software-ontwikkelafdeling binnen Philips (figuur 5). Zoals blijkt zijn hier andere oorzaken genoemd en zullen dus andere verbeteringsacties vereist zijn.

Nadat op bovengenoemde, relatief eenvoudige wijze is aangetoond hoe omvangrijk de verschillen zijn tussen plan en werkelijkheid en men inzicht heeft gekregen in de oorzaken van eventuele verschillen, wordt de volgende vraag



Figuur 4: Genoemde oorzaaksoorten m.b.t. overschrijding doorlooptijd



Figuur 5: Oorzaaksoorten genoemd op een andere Philipsafdeling m.b.t. overschrijding doorlooptijd (Van Genuchten (1991))

actueel: welke acties c.q. verbeteringen moet men doorvoeren om deze oorzaken weg te nemen of in ieder geval het effect ervan te reduceren?
In veel gevallen kunnen, indien de oorzaak van een probleem bekend is, passende maatregelen uitgevoerd worden. Wanneer maatregelen niet voor de hand liggen, dan zullen op afdelingsniveau bijeenkomsten moeten worden georganiseerd om tot verdere stappen te kunnen komen. Een voorbeeld van een beheersmaatregel die genomen werd in een situatie met een doorlooptijdprobleem, veroorzaakt door projectgebonden oorzaken, zullen we hier ter verduidelijking bespreken.

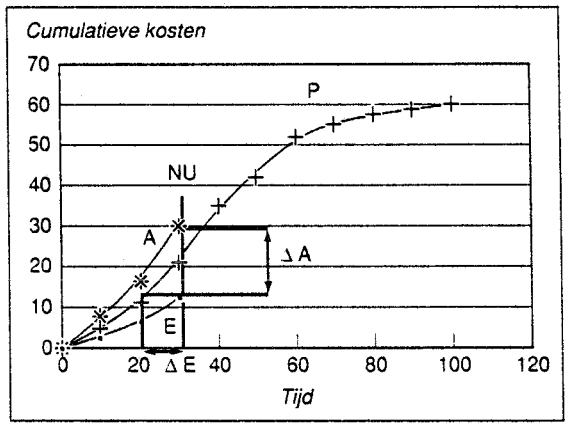
Bij een afdeling binnen Philips waar systeemsoftware wordt ontwikkeld, zijn gegevens volgens de bovenbeschreven aanpak verzameld en geordend. Tijdens een bijeenkomst met de betreffende projectleiders zijn de gegevens gepresenteerd en nader geanalyseerd. Er werd geconstateerd dat de voortgang van de ontwikkelactiviteiten werd verstoord door onderhoud. Hierdoor werd de geplande doorlooptijd in realiteit fors overschreden. De verbeteringsmaatregel die tijdens die bijeenkomst werd bedacht en naderhand werd doorgevoerd, had betrekking op het clusteren van onderhoud in aparte onderhoudsweken. Voor een uitgebreidere beschrijving van deze situatie verwijzen we naar (Van Genuchten (1991)). De eerste resultaten van deze maatregel zijn veelbelovend.

Een belangrijk uitgangspunt bij het meten aan softwareontwikkeling en het formuleren van verbeteringsmaatregelen is de betrokkenheid van de projectmedewerkers. Zij moeten overtuigd zijn van het nut van de metingen en bereid zijn tijd en energie te steken in het leveren van betrouwbare meetgegevens. Een goede voorlichting over het waarom en hoe van de metingen is belangrijk. Meetresultaten moeten niet worden gebruikt voor bijvoorbeeld produktiviteitsmetingen van individuele ontwikkelaars. Zij dienen uitsluitend om inzicht te krijgen in het proces van software-ontwikkeling zodat de betreffende afdeling zelf, i.c. de medewerkers, passende verbeteringsmaatregelen kunnen formuleren. Verbeteringsmaatregelen die aldus ontstaan, worden door de afdeling gedragen en zullen meer kans van slagen hebben dan maatregelen die over de hoofden van de betrokkenen tot stand zijn gekomen. Indien aan bovengenoemde voorwaarden wordt voldaan dan zullen meetresultaten motiveren en de betrokkenheid vergroten.

3 Het analyseren van een project

De analyse van software-ontwikkeling op een afdeling is in de vorige paragraaf besproken. Het doel van deze analyse was een betere beheersing van de ontwikkeling op afdelingsniveau. In deze paragraaf wordt besproken hoe dezelfde analysetechniek kan worden gebruikt om een lopend project te bewaken. Het bewaken van een project bestaat ondermeer uit het vroegtijdig signaleren van verschillen tussen plan en werkelijkheid. Een veelvoorkomende klacht van klanten van software-ontwikkelafdelingen zou als volgt kunnen worden geformuleerd: '*Dat de software weer eens te laat is, is al vervelend genoeg. Maar dat ik pas kort voor oplevering weet dat er vertraging is en de omvang ervan weet, maakt het probleem vele malen groter*'. Deze paragraaf zal bespreken hoe de gegevens die tijdens het project worden verzameld, kunnen bijdragen aan een betere bewaking van software-ontwikkeling.

De analyse maakt gebruik van de S-curve, een techniek die al vele tientallen jaren wordt toegepast en bijvoorbeeld wordt beschreven door Harrison (1977). De S-curve maakt een vergelijking van geplande en werkelijke uitvoering van een project mogelijk. In de curve worden de cumulatieve mensuren uitgezet tegen de tijd. Zij neemt gewoonlijk de vorm aan van een S omdat een project vaak als volgt verloopt: gestart wordt met een beperkte groep medewerkers, na de start volgt een periode van grote activiteit en het project eindigt wederom met een kleine groep medewerkers. Toegepast op software-ontwikkeling: de specificaties worden door een klein aantal ontwikkelaars opgesteld, het ontwerp en het coderen gebeurt met een grotere groep ontwikkelaars, terwijl de integratietest en de installatie door een klein aantal medewerkers gebeurt. Een voorbeeld van een S-curve is afgebeeld in figuur 6.



Figuur 6: Een voorbeeld van een S-curve

De S-curve bestaat uit drie lijnen: de *P*-lijn (*Planned*), de *A*-lijn (*Actual*) en de *E*-lijn (*Earned value*). We gebruiken de Engelse benamingen van de lijnen omdat de begrippen *P*, *A* en *E* lijn bekende begrippen zijn. De betekenis van de drie lijnen zal achtereenvolgens worden toegelicht.

- De *P*-lijn. Deze lijn beschrijft de cumulatieve geplande mensuren tegen de geplande einddatum van de activi-

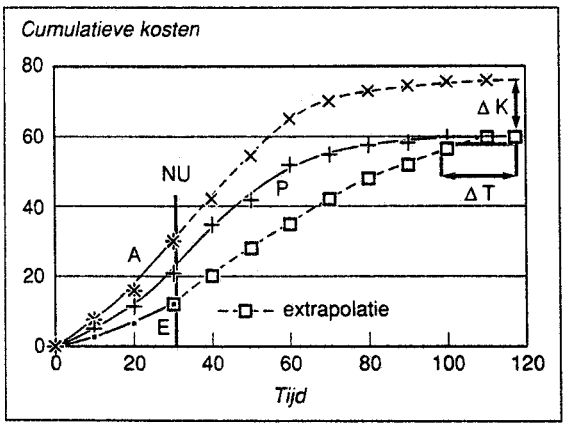
teiten. De lijn is eenvoudig af te leiden uit het projectplan. De benodigde gegevens zijn verzameld volgens de werkwijze beschreven in paragraaf 2.

- De *A*-lijn. Deze lijn beschrijft de cumulatieve werkelijke uren tegen de werkelijke einddatum van de activiteiten. Deze lijn is eenvoudig af te leiden uit de registratie van de voortgang van het project. Ook deze gegevens zijn verkregen volgens de eerder beschreven werkwijze.
- De *E*-lijn. De *P*-lijn en de *A*-lijn zijn niet rechtstreeks met elkaar te vergelijken omdat in een project vaak zowel de geplande en werkelijke doorlooptijd als de geplande en werkelijke inspanning verschillen. De *P*- en *A*-lijn verschillen dus ten opzichte van twee assen. Daarom is een derde lijn nodig: de *E*-lijn beschrijft de cumulatieve geplande (begrote) uren van de uitgevoerde activiteiten.

De *P*-, *A*- en *E*-lijn kunnen nu twee aan twee met elkaar worden vergeleken. Het verticale verschil tussen de *A*-lijn (werkelijke uren tegen werkelijke einddatum) en de *E*-lijn (geplande uren tegen werkelijke einddatum) geeft het *verschil in mensuren*, het inspanningsverschil tussen plan en werkelijkheid. Het verticale verschil is in figuur 6 aangegeven met ΔA. Het horizontale verschil tussen de *P*-lijn (geplande uren tegen geplande einddatum) en de *E*-lijn (geplande uren tegen werkelijke einddatum) geeft het *verschil in doorlooptijd* tussen plan en werkelijkheid. Het horizontale verschil is aangegeven met ΔE. Tot zover de behandeling van de traditionele S-curve.

We gaan nu in op de extrapolatie van de S-curve. Een voorbeeld van een geëxtrapoleerde S-curve staat in figuur 7.

Het is interessant te constateren dat er verschillen zijn opgetreden tussen plan en werkelijkheid. Het is belangrijker te voorspellen wat de tot nu toe geconstateerde verschillen



Figuur 7: Voorbeeld van een geëxtrapoleerde S-curve

betekenen voor de uitloop van het totale project. Normaal gesproken wordt de tot nu geconstateerde uitloop geëxtrapoleerd voor de rest van het project. Een voorbeeld: in de specificatiefase hebben we een overschrijding van de doorlooptijd van 20 procent vastgesteld. Extrapolatie betekent dat de uitloop voor het totale project 20% zal zijn.

Wij stellen voor het inzicht in de oorzaken van uitloop te gebruiken bij het extrapoleren van de S-curve. Het is namelijk niet alleen bekend hoeveel het project is uitgelopen, maar ook waarom het project is uitgelopen. De oorzaken zijn immers vastgelegd. De projectleiders kunnen de invloed van de geconstateerde oorzaken van uitloop schatten voor de rest van het project. Dit heeft twee voordelen: ten eerste wordt het inzicht van de projectleiders benut bij de extrapolatie. Ten tweede zijn de projectleiders betrokken bij de extrapolatie. Het is niet een buitenstaander die vertelt hoe ver hun project zal gaan uitlopen. Zij voorspellen het in feite zelf.

Een voorbeeld: de specificatiefase van een project overschrijdt de geplande doorlooptijd met 60%. Capaciteitsgebonden oorzaken zijn verantwoordelijk voor 50% van de uitloop. De projectleiders menen dat dit alleen in de eerste fase voor kan komen omdat mensen uit eerdere projecten losgeweekt moesten worden. Zij schatten dat de overschrijding van de doorlooptijd ten gevolge van capaciteitsgebonden oorzaken terugloopt naar 0%. Stel dat de invloed van de andere oorzaken volgens de projectleiders gelijk blijft, dan is de verwachte overschrijding van de doorlooptijd in de rest van het project 30%.

De gevolgen van de oorzaken in de rest van het project kunnen op deze manier door de projectleiders worden geschat. De overschrijding van de geplande inspanning in de rest van het project kan op een vergelijkbare manier worden geraamd. We hebben deze manier van extrapoleren toegepast op een aantal projecten. Het succes dient ons inziens in de meeste gevallen niet afgeleid te worden van de nauwkeurigheid van de extrapolatie, maar van de reactie op de extrapolatie. Wij hebben ervaren dat de discussies binnen het projectteam over verschillen tussen plan en werkelijkheid, over oorzaken hiervan en mogelijke maatregelen bijzonder constructief en leerzaam zijn voor het functioneren van het team. Dit wordt versterkt door het feit dat er al in een vroeg stadium duidelijke signalen zijn aan de hand waarvan gerichte sturing mogelijk is. Wij hebben bovendien ervaren dat het succes van het gebruik van de S-curve voor een belangrijk deel wordt bepaald door de betrokkenheid van het projectteam. Het team moet geloven in de ontwikkelingen die door de curves worden aangegeven.

4 Conclusies en aanbevelingen

In dit artikel is beschreven hoe het verzamelen van gegevens op redelijk korte termijn tot verhoging van het inzicht in software-ontwikkeling kan leiden. De medewerking van de ontwikkelaars blijkt van doorslaggevend belang te zijn bij het verzamelen van gegevens op de voorgestelde manier. In het onderzoek is gebleken dat de medewerking om een aantal redenen noodzakelijk is.

Ten eerste is de medewerking noodzakelijk voor het verkrijgen van *nauwkeurige informatie* over het verloop van het project.
Ten tweede is de medewerking noodzakelijk voor het *bepalen van de oorzaken* van verschillen tussen plan en werkelijkheid en om de verwachte invloed van de genoemde oorzaken op de rest van het project te schatten. Ten derde zal de *acceptatie* van de maatregelen om de beheersing te verbeteren groter zijn als de ontwikkelaars bij het bedenken van de maatregelen betrokken zijn. Dit geldt ook voor de acceptatie van de extrapolatie van de S-curve.

Inmiddels hebben wij een ruime ervaring opgedaan met het verzamelen van gegevens op de voorgestelde wijze. Behalve op de afdeling waar dit onderzoek is uitgevoerd, zijn ook metingen verricht op een aantal andere afdelingen binnen én buiten Philips. In totaal beschikken we inmiddels over meetgegevens van meer dan 500 activiteiten wat neer komt op circa 30.000 mensuren. Deze ervaring brengt ons tot de volgende conclusies en aanbevelingen:

- Met een relatief geringe inspanning is het mogelijk gegevens over software-ontwikkeling te verzamelen waarop verbeteringsmaatregelen gebaseerd kunnen worden.
- De betrokken ontwikkelaars en projectleiders blijken de verkregen informatie op prijs te stellen.
- We willen uit eigen ervaring benadrukken dat software-ontwikkeling op verschillende plaatsen zo sterk verschilt, dat het trekken van conclusies op basis van elders verzamelde, empirische gegevens moet worden afgeraden. Organisaties doen er ons inziens verstandig aan in de eigen organisatie gegevens over oorzaken van verschillen tussen plan en werkelijkheid te verzamelen, om op basis van het verkregen inzicht adequate en op de eigen organisatie toegesneden maatregelen te kunnen nemen.
- Het is mogelijk gebleken met de verzamelde gegevens in combinatie met de S-curve de voortgang van het project op een voor de betrokkenen overtuigende wijze te extrapoleren. Dit had in één geval als resultaat dat voor een dreigende uitloop gewaarschuwd kon worden nog vóór de helft van het project verstreken was.

Zoals in de inleiding al is aangegeven, is het verzamelen van gegevens over software-ontwikkeling een voorwaarde

Oorzaken van verschillen tussen plan en werkelijkheid

Capaciteitsgebonden oorzaken
11 Capaciteit niet aanwezig door uitlopen voorgaande activiteit (binnen hetzelfde project)
12 Capaciteit aanwezig door korter duren voorgaande activiteit (binnen hetzelfde project)
13 Meer tijd aan ander werk besteed dan gepland (in een ander project)
19 Overig

Personeelsgebonden oorzaken
21 Te weinig ervaring met ontwikkelomgeving
22 Meer niet ingewerkte mensen ingezet dan gepland
23 Leereffect
29 Overig

Niet voldaan aan inputvoorwaarden
31 Userrequirements te laat
32 Userrequirements van onvoldoende kwaliteit
33 Softwarerequirements te laat
34 Softwarerequirements van onvoldoende kwaliteit
35 Design te laat
36 Design van onvoldoende kwaliteit
39 Overig

Produktgebonden oorzaken
41 Wijziging van requirements tijdens activiteit
42 Functionele specificaties technisch niet mogelijk
43 Complexiteit van de toepassing werd onderschat
44 Meer problemen door performance-eisen en geheugenbeperkingen dan verwacht
45 Produkt van onvoldoende kwaliteit of te beperkte functionaliteit ontwikkeld (redesign nodig)
46 Meer fouten ontdekt bij testen dan verwacht
47 Hoeveelheid werk van toepassing onderschat
48 Hoeveelheid werk van toepassing overschat
49 Overig

Organisatorische oorzaken
51 Minder continuïteit in projectbezetting dan gepland
52 Meer werkonderbrekingen dan gepland
53 Inzichtverschillen tussen projectmedewerkers
54 Afwijken van in plan voorgeschreven werkwijze
59 Overig

Middelengebonden oorzaken
61 Ontwikkelapp. te laat of te weinig beschikbaar
62 Ontwikkeltools te laat of te weinig beschikbaar
69 Overig

Overig
71 t/m *79* Overig

voor het verbeteren van de beheersing. We hopen hiervoor in dit artikel een bijdrage geleverd te hebben door te laten zien hoe het mogelijk is op een eenvoudige manier te starten met het verzamelen van gegevens.

Het is duidelijk dat het beschreven meetinstrument nog lang niet voorziet in alle informatie die nodig is voor het beheersen van softwareprojecten. Men hoeft alleen maar te denken aan kwaliteitsmetrieken, kengetallen voor de ontwikkeling en gegevens over herbruikbare softwarecomponenten. Het specificeren en ontwikkelen van een informatiesysteem voor het beheersen van software-ontwikkeling is het doel van een lopend onderzoek aan de Technische Universiteit in Eindhoven. Het beschreven meetinstrument zien we als een eerste, maar niet onbelangrijke stap naar een betere informatievoorziening voor het beheersen van softwareprojecten.

Referenties

Bemelmans, T. (1989), 'Informatiekunde: vragen, geen antwoorden', in: *Jubileumnummer Informatie*, jaargang 31, blz. 447-456.

Brooks, F.B. (1975), *The Mythical ManMonth, Essays on Software Engineering*, Addison Wesley Publishing Company, London.

Brooks, F.B. (1987), 'No silver bullet, essence and accidents of software engineering', in: *IEEE Computer*, april 1987.

Genuchten, M. van, (1991), *Towards a software factory*, Proefschrift TU Eindhoven, te verschijnen.

Gilb, T. (1988), *Principles of software engineering management*, Addison Wesley Publishing Company, London

Harrison, F.L. (1977), *Advanced project management*, Gower Publishing company, Aldershot.

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

Wijnen, G., P. Storm en W. Renes (1984), *Projectmatig werken*, Marka paperback reeks, Uitgeverij Het Spectrum, Utrecht.

Ir. Frans van Lierop en Ir. Robert Volkers zijn als A.I.O. verbonden aan de Technische Universiteit in Eindhoven. Zij ronden binnenkort hun tweede fase opleiding 'Ontwerpen van Logistieke Besturingssystemen' af.

Ir. Michiel van Genuchten werkt als adviseur bij de Lighthouse Consulting Group. Hij is tevens verbonden aan de vakgroep Bestuurlijke Informatiesystemen en Automatisering van de afdeling Technische Bedrijfskunde van de Technische Universiteit in Eindhoven.

Dr. ir. Fred Heemstra is als universitair hoofddocent verbonden aan de vakgroep Bestuurlijke Informatiesystemen en Automatisering van de afdeling Technische Bedrijfskunde van de Technische Universiteit in Eindhoven.