Begroten van software-projecten ten behoeve van de procesautomatisering

J. Akkerman

1.	Inleiding	P2010– 3
1.1.	Stand van zaken	P2010– 3
1.2.	Aanbevelingen	P2010– 4
2.	Informatieverwerking bij de procesindustrie	P2010– 5
3.	Ontwikkelingstraject van een informatieverwerkend systeem	P2010– 8
3.1.	Haalbaarheidsstudie	P2010– 9
3.2.	Functionele specificatie	P2010– 9
3.3.	Technisch ontwerp	P2010–10
3.4.	Realisatie	P2010–11
3.5.	Gebruik en beheer	P2010–11
3.6.	Projectbeheersingsmethoden	P2010–12
4.	Beïnvloedingsfactoren	P2010–17
5.	Begrotingsmodellen	P2010–20
5.1.	Werkwijze voor het opstellen van een begroting	P2010–21
5.2.	Parametrische modellen	P2010–24
5.2.1.	COCOMO	P2010–24
5.2.2.	PUTNAM	P2010–27
5.2.3.	FPA	P2010–29
5.3.	Analoge model	P2010–34
6.	Literatuur	P2010–37

1. Inleiding

In toenemende mate worden cost engineers geconfronteerd met projecten waarvan „automatisering" deel uitmaakt. Voor niet terzake kundigen wordt met name de software-ontwikkeling als abstract en moeilijk begrootbaar en beheersbaar ervaren, daarin geruggesteund door regelmatig terugkerende publikaties in de pers die melding maken van automatiseringsprojecten die te lang duren en het budget drastisch overschrijden. Alhoewel deze publikaties meestal over projecten gaan waarbij de overheid betrokken is wil dit niet zeggen dat hetzelfde niet bij het bedrijfsleven gebeurt. Het verschil is dat het bedrijfsleven geen Rekenkamer heeft die deze voorvallen in de openbaarheid brengt.
Het is een illusie te denken dat in deze bijdrage aan het handboek voor cost engineers een begrotingsmethode kan worden geboden waarmee budget- en levertijdoverschrijdingen tot het verleden behoren; daarvoor zijn de publikaties in de pers, waarbij gerenommeerde software-bouwers betrokken zijn, te talrijk.
Wel wordt getracht duidelijk te maken welke facetten bij de software-ontwikkeling een rol spelen en op welke wijze de risico's beperkt kunnen worden.
Bij het schrijven van dit document is er vanuit gegaan dat de lezer geen kennis heeft van de werkwijze bij het realiseren van een software-project. In de paragrafen 2 en 3 worden een aantal begrippen geïntroduceerd.
In deze bijdrage wordt de nadruk gelegd op het begroten van software-ontwikkeling ten behoeve van de proces-automatisering.
Andere vormen van automatisering (zoals bijv. kantoorautomatisering, CAD-systemen etc.) worden buiten beschouwing gelaten, evenals de wijze waarop de hardware-keuze tot stand komt.

1.1. Stand van zaken

Enquêtes tonen aan dat er weinig verschil is tussen de stand van zaken in Amerika en Nederland ten aanzien van het begroten en het beheersen van software-ontwikkeling. Wanneer we er vanuit gaan dat de enquêtes representatief zijn dan kunnen we zeggen dat:

– circa 1/3 van de organisaties geen begroting maakt voor software-ontwikkeling; van de organisaties die wel een begroting maken doet 60 procent dit op basis van intuïtie en ervaring; 16 procent maakt gebruik van een formele begrotingsmethode;
– circa 1/2 houdt geen registratie bij over de software-ontwikkeling en voert geen nacalculaties uit;
– circa 4/5 van de projecten hebben te kampen met budget- en

levertijdoverschrijdingen; deze bedraagt zowel voor budget- als levertijd gemiddeld 50 procent.

1.2. Aanbevelingen

Op grond van de resultaten van bovengenoemde enquête en eigen ervaring leidt dit tot de volgende aanbevelingen:

– Leg de omvang van het project vast

Baseer de begroting op een duidelijke specificatie van de te leveren functionaliteit en de te verrichten activiteiten: maken opleidingen (voor gebruikers en/of onderhoudsgroep), acceptatietest en invoering deel uit van het project of wordt dit op een andere wijze geregeld.

– Kies een begrotingsmodel dat aansluit bij de werkwijze van de organisatie

Bijvoorbeeld indien de teambezetting tijdens de bouwtijd van het project constant gehouden wordt is het niet zinvol om te begroten met het Putnam-model waarbij er vanuit gegaan wordt dat de bezetting van het bouwteam de Rayleigh-verdeling volgt.

– Maak gebruik van de gegevens van eerder gerealiseerde projecten en hou ook zelf een registratie bij.

– Gebruik gezond verstand

Vertrouw niet blindelings op het begrotingsmodel maar toets de resultaten aan de hand van gerealiseerde projecten of aan de hand van resultaten verkregen uit een ander begrotingsmodel.

– Beperk de omvang en de doorlooptijd van de projecten

Deel eventueel grote projecten op in kleinere deelprojecten. Dit verbetert de beheersbaarheid. Bij projecten met een te lange looptijd is het moeilijk de teamleden gemotiveerd te houden.

– Evalueer

Evalueer op gezette tijden de stand van zaken binnen het project en stel de begroting bij indien daar aanleiding toe bestaat. Weliswaar vermindert dit niet de kans op budgetoverschrijdingen, maar wel de kans op doorlooptijd-overschrijdingen. Het is nagenoeg onmogelijk in een laat stadium hiertoe maatregelen te treffen. Brooks waarschuwt in dit verband: *„Adding manpower to a late software project makes it later”*.

– Evolueer

De factoren die medebepalend zijn voor de benodigde inspanning wijzigen in de loop der tijd bijvoorbeeld door opleidingen neemt het kennisniveau van de software-ontwikkelaars toe, de introductie van nieuwe programmeertalen leidt tot een hogere produktiviteit etc. Het gebruikte begrotingsmodel moet hieraan worden aangepast.

– Informeer de bedrijfsleiding

Het belang van het schatten van de software-projecten moet ook door de bedrijfsleiding worden onderkend. Dit wordt gestimuleerd door regelmatig de bedrijfsleiding te informeren van de stand van zaken.

– Betrek de eindgebruikers in een vroeg stadium bij de ontwikkeling van het systeem

Zij kunnen vanuit hun ervaring het systeem toetsen op bruikbaarheid in praktijksituaties. Daarnaast zullen zij zich medeverantwoordelijk voelen (en ook zijn!) voor een goed eindresultaat.

2. Informatieverwerking bij de procesindustrie

In het traject tussen orderbinnenkomst en orderuitlevering vindt een groot aantal handelingen plaats waarbij informatieverwerkende systemen worden ingezet. Met informatieverwerkend systeem worden niet alleen de geautomatiseerde (computer)systemen bedoeld; het is het geheel van handmatige en geautomatiseerde procedures.
Als we bij normaal (ongestoord) produktieverloop een klantorder volgen dan zien we dat deze op onderstaande wijze verwerkt wordt.

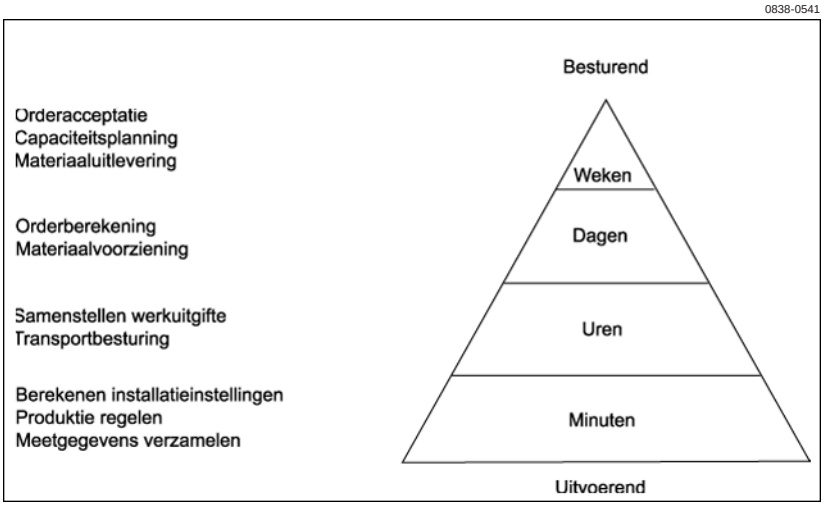
1. Aan de hand van een aantal technische acceptatie criteria (gewenste kwaliteit, dimensies etc.) wordt bepaald of het bestelde produkt kan worden geproduceerd.
2. Met behulp van het capaciteitsplan wordt bepaald of de gewenste levertijd kan worden gerealiseerd.
3. Bij de produktieplanning worden de posten van de orders zodanig samengevoegd dat gedurende optimale tijd met de produktielijn hetzelfde produkt kan worden geproduceerd.
4. De benodigde grondstoffen worden tijdig besteld of, indien deze al aanwezig zijn, gereserveerd voor deze order.
5. Enige tijd voordat de produktie moet plaatsvinden wordt aan de producerende afdeling een werkuitgifte gezonden, waarin staat aangegeven welk produkt moet worden geproduceerd.

6. Aan de hand van de werkuitgifte wordt opdracht gegeven de grondstoffen uit het magazijn aan te voeren.
7. De installatie-instellingen (Engels: setpoints, Duits: Sollwerte) worden berekend waarmee het produkt vervaardigd gaat worden.
8. Vervolgens wordt de installatie volgens deze gegevens ingesteld en vindt de produktie plaats.
9. Tijdens het produktieverloop worden metingen uitgevoerd en monsters genomen en geanalyseerd om de kwaliteit te waarborgen. De gegevens van deze metingen (op tijd- en/of produktbasis) worden opgeslagen en dienen als basis voor rapporteringen, statistische analyses en eventueel als naslag voor het geval de klant reclameert.
   Wanneer het produkt gereed is wordt het gereedgemeld en opgeslagen.
10. Na de gereedmelding wordt zorggedragen voor uitlevering aan de klant en voor facturering.

Niet altijd zal het produktieproces zo ongestoord verlopen als hierboven beschreven. Het kan bijvoorbeeld gebeuren dat er een defect optreedt in de produktielijn waardoor bepaalde bewerkingen tijdelijk niet meer mogelijk zijn en het produkt dus niet volgens klantenspecificaties geproduceerd kan worden; de grondstoffen kunnen van een andere kwaliteit zijn dan op basis van de steekproeven vooraf werd aangenomen etc. Daarom moet het informatie verwerkende systeem ook voorzien in een oplossing voor (reëel mogelijke) verstoringen die op kunnen treden.
Er is een bepaalde hiërarchie te ontdekken in de hierboven beschreven functies:

1. orderacceptatie;
2. capaciteitsplanning;
3. orderberekening;
4. materiaalvoorziening;
5. samenstellen werkuitgifte;
6. transportbesturing;
7. berekenen installatie-instellingen;
8. produktie regelen;
9. meetgegevens verzamelen;
10. materiaaluitlevering en facturering.

Deze hiërarchie kan door middel van een piramide weergegeven worden (zie fig. 1).



Figuur 1.

Aan de top van de piramide vinden de besturende activiteiten plaats, aan de voet van de piramide de uitvoerende activiteiten.
Deze piramide kan in een aantal lagen worden onderverdeeld. In dit voorbeeld is gekozen voor een vierlagen structuur. Aantal lagen en de wijze waarop deze genummerd worden (van boven naar beneden of omgekeerd) is niet voor alle bedrijven hetzelfde en hangen samen met de omvang en de organisatie van het bedrijf.
De overeenkomst tussen de lagen in deze structuur is dat elke laag (behalve de bovenste) van de bovenliggende laag een opdracht ontvangt en gegevens terugmeldt over ontvangen opdrachten. Verder zijn er omstandigheden van buitenaf (storingen) die het functioneren van de laag beïnvloeden.
Verschillen tussen de lagen hebben betrekking op de geldigheidsduur van de gegevens, de omvang van het besturingsgebied en de inhoud van opdracht en rapportage.
Van boven naar beneden in de piramide zal de geldigheidsduur van de gegevens in het systeem afnemen, zullen de opdrachten vaker worden verstuurd, meer gestructureerd en gedetailleerd zijn. Voor rapporten geldt hetzelfde.
In dit licht gezien gedraagt elke laag zich als een regelkring: de opdracht is de gewenste waarde, de rapportage de gerealiseerde waarde en de omstandigheden van buitenaf de verstoringen. De vereiste regelsnelheid neemt van top naar voet van de piramide toe: er moet sneller op verstoringen in de omgeving gereageerd worden en de gegevens zijn korter geldig. Om deze reden is het noodzakelijk voor de

systemen bij de voet van de piramide real-time-systemen toe te passen; deze systemen reageren onmiddellijk op gebeurtenissen in de omgeving. Een voorbeeld is het stoppen van de aanvoer van grondstoffen wanneer verderop in het produktieproces een stagnatie optreedt.
Het tegenovergestelde van de real-time-systemen zijn de batch systemen; hierbij wordt een bepaalde taak uitgevoerd met tevoren gespecificeerde invoergegevens. Een voorbeeld hiervan is het onderzoeken van de nieuw ontvangen klantenorders op gelijksoortige orderposten en het combineren hiervan tot produktieposten. Typisch hoeft dit slechts een tot enkele malen per dag te worden uitgevoerd. Het is mogelijk hiervoor een real-time-systeem toe te passen; noodzakelijk is dit niet.
De lagenstructuur zegt op zich niets over de wijze waarop dit hardware-technisch wordt gerealiseerd. Dit is ook sterk afhankelijk van de omvang van het bedrijf en de complexiteit van het produktieproces. Het is theoretisch mogelijk alle functies met een computer te realiseren. In het algemeen echter worden meerdere gekoppelde computers gebruikt om zo'n lagenstructuur te realiseren, waarbij elke computer een aantal van de genoemde functies voor zijn rekening neemt.
Door de voortschrijding van de techniek worden de computers steeds krachtiger, waardoor ook de verdeling van de functionaliteit over de computers wijzigt.

3. Ontwikkelingstraject van een informatieverwerkend systeem

Elk produkt, dus ook een informatieverwerkend systeem, heeft een bepaalde levenscyclus. De levenscyclus vangt aan wanneer de eerste gedachte ontstaan om te komen tot het produkt. Vervolgens wordt een haalbaarheidsstudie uitgevoerd om te onderzoeken of het op de markt brengen van het produkt zinvol is. Het criterium voor zinvol is meestal dat de baten de kosten moeten overtreffen. Ook strategische overwegingen kunnen hierbij een rol spelen.
Als besloten wordt het produkt verder te ontwikkelen dan wordt een specificatie opgesteld, de specificatie wordt uitgewerkt tot een ontwerp, het produkt wordt vervaardigd en op de markt gebracht. Na verloop van tijd veranderen de eisen die de markt aan het produkt stelt en herhaalt de cyclus zich: een nieuw produkt wordt ontwikkeld en vervangt de oude versie.
In feite heeft een informatieverwerkend systeem eenzelfde levenscyclus. Deze wordt hieronder beschreven.

3.1. Haalbaarheidsstudie

Tijdens de haalbaarheidsstudie wordt beknopt beschreven welke functionaliteit de gebruiker van het systeem verwacht, welke baten het systeem oplevert en wat het systeem gaat kosten.
Als de baten tijdens de levensduur van het systeem de kosten overtreffen of wanneer de investering op strategische gronden voordelen biedt zal de bedrijfsleiding in het algemeen besluiten over te gaan tot een nadere specificatie van het systeem, de functionele specificatie.

3.2. Functionele specificatie

Tijdens de haalbaarheidsstudie is de functionaliteit zeer globaal beschreven en zal de bepaling van de kosten en baten ook ruime toleranties hebben. Om deze toleranties te verkleinen moet de functionaliteit met meer detail worden beschreven. Dit vindt plaats in een functionele specificatie.
Echter, specificeren is een tijdrovende bezigheid, die door hooggekwalificeerd en dus duur personeel wordt uitgevoerd. Indien de specificatie onmiddellijk tot in detail zou worden opgesteld bestaat het risico dat dan pas zou blijken dat het rendement van het project tegenvalt. Om deze reden is de tendens eerst een globale functionele specificatie op te stellen met bijbehorende kosten en baten en deze, na goedkeuring door de bedrijfsleiding, uit te werken tot een gedetailleerde functionele specificatie.
De verschillen tussen de globale en de gedetailleerde specificatie zijn dus in hoofdzaak de mate van detail en het doel van de specificatie. Het doel van de globale functionele specificatie is om te dienen als basis voor besluitvorming over het wel of niet doorgaan op de ingeslagen weg, het doel van de gedetailleerde functionele specificatie is om voldoende informatie aan de ontwerpers te verschaffen om het systeem te kunnen realiseren.
De overeenkomst tussen globale en gedetailleerde functionele specificatie is dat beide beschrijven *wat* het systeem volgens de gebruikers moet doen en zich niet bezighouden met de vraag *hoe* dit moet gebeuren.
In de functionele specificatie worden dus de eisen en de wensen van de gebruikers vastgelegd, de verantwoordelijkheid dat dit juist en volledig gebeurt ligt dan ook bij de gebruikers („You get what you ask, not what you want”).
De functionele specificatie en met name de gedetailleerde functionele specificatie is het grensvlak tussen gebruikers enerzijds en de ontwerpers anderzijds. Het document moet daarom voor beide disciplines bruikbaar zijn.

Voor de gebruiker betekent dit in het algemeen tekst, aangevuld met tekeningen en schema's, de ontwerpers geven de voorkeur aan een specificatie waarin de gegevens en de functies op een bepaalde manier wordt gestructureerd.
De gegevens worden samengenomen tot groepen gegevens die logischerwijs bij elkaar behoren bijvoorbeeld gegevens die betrekking hebben op het produkt, gegevens die betrekking hebben op een wacht, gegevens die betrekking hebben op een installatiedeel etc. Deze groepen heten genormaliseerde logische gegevensgroepen of entiteiten, het proces van het groeperen van de gegevens heet normaliseren. Dit normaliseren gebeurt in een aantal stappen, wanneer gesproken wordt van bijvoorbeeld de 3e normaalvorm dan betekent dit dat de eerste 3 normaliseringsstappen zijn uitgevoerd.
Een entiteit is dus in feite niets anders dan een opsomming van welke gegevens voor een produkt, wacht, installatiedeel etc. moeten worden vastgelegd. De gegevens die deel uitmaken van een entiteit worden attributen genoemd.
Een functie is een stuk(je) software dat gegevens verwerkt tot andere gegevens. Een heel eenvoudig voorbeeld is het berekenen van een snelheid: uit de gegevens afgelegde weg en verstreken tijd wordt de snelheid bepaald.
De functies die in de functionele specificatie worden beschreven zijn in het algemeen veel complexer. Deze complexe functies worden onderverdeeld in een aantal niveaus, waarbij in de onderliggende niveaus steeds meer details zichtbaar worden.
Ter illustratie een analogie met een landkaart. Wanneer we de beschrijving van het systeem vergelijken met de kaart van een werelddeel, dan is dit werelddeel (het systeem) weer onderverdeeld in landen (de subsystemen). Om meer van een bepaald land (subsysteem) te weten te komen wordt de kaart van het land erbij gehaald. Vervolgens de kaart van de provincie, streek etc. De niveaus wereldkaart , landkaart etc. worden ook aangebracht in de functionele specificatie en heten daar systeemdiagram, subsysteem, etc. Dit proces van herhaald opsplitsen van complexe functies in eenvoudiger functies heet functionele decompositie.
Het aantal niveaus hangt af van de complexiteit van het systeem. Op het allerlaagste niveau vinden we de primitieve functies (ook wel elementaire functies genoemd), dit zijn functies die niet verder op te splitsen zijn.

3.3. Technisch ontwerp

Tijdens het functionele ontwerp is vastgelegd wat het systeem moet doen, tijdens het technische ontwerp wordt bepaald hoe dit gereali-

seerd kan worden. Hiertoe wordt de geschikte hardware gekozen en wordt de functionele specificatie vertaald naar programmastructuren en een gegevensopslag structuur. Deze „technische" structuren kunnen afwijken van de structuren in de functionele specificaties.

3.4. Realisatie

Tijdens de realisatie wordt het technisch ontwerp vertaald naar de gekozen programmeertaal, het programmeren of coderen. Zoals een boek bestaat uit zinnen, zo bestaat een programma uit regels programmeertaal, ook wel „(source-)statements" of „lines of (source-)code" genoemd.
Afhankelijk van de omvang kan het systeem uit een tot honderden afzonderlijke programma's bestaan.
In de functionele specificatie is bepaald welke entiteiten er zijn, dus welke gegevens vastgelegd moeten worden. Dit wordt gerealiseerd in een database. Een database is in feite niets anders dan een verzameling tabellen.
Een voorbeeld van een eenvoudige database is een adressenlijst van personen. De adressenlijst begint met de kopregel „Naam, Adres, Woonplaats en Telefoonnummer". De entiteit „Persoon" bevat dus de gegevens naam, adres, woonplaats en telefoonnummer. Hieronder volgen een aantal regels waarop van de personen de feitelijke naam, adres etc. staan. Zo'n regel heet een occurrence, er zijn dus evenveel occurrences als er personen in de adressenlijst staan.
Ook het testen van alle programma's, afzonderlijk en in samenhang met elkaar, maakt deel uit van de realisatie.
De realisatie wordt afgesloten met een acceptatietest, waarbij de gebruiker verifieert dat de bestelde functionaliteit inderdaad geleverd is. Na een goedgekeurde acceptatietest wordt het systeem in bedrijf genomen.

3.5. Gebruik en beheer

Na verloop van tijd ontstaan als gevolg van een veranderende organisatie de behoefte om de systemen te wijzigen. Wanneer het binnen het huidige systeem niet meer mogelijk is om de wensen te honoreren onstaat de gedachte om te komen tot een nieuw informatieverwerkend systeem, waarmee de cyclus gesloten is.
De opeenvolgende stappen in het automatiseringstraject kenmerken zich door een steeds vaster omlijnd beeld van wat het systeem moet doen, wat de kosten en de baten zijn en hoe lang de ontwikkeltijd is. In de opeenvolgende stappen is het dan ook mogelijk een steeds minder onnauwkeurige begroting te maken. Ter indicatie de volgende nauwkeurigheden:

Haalbaarheidsstudie	25%
Globale functionele specificatie	15%
Gedetailleerde functionele specificatie	10%
Technisch ontwerp	5%

Als we de ontwikkeling van het systeem vergelijken met een piramide dan wordt deze van de top af benaderd en dan naar de basis van de piramide toe in steeds meer details zichtbaar. Dit wordt dan ook wel de top-down aanpak genoemd.
De hierboven beschreven levenscyclus heeft niet noodzakelijkerwijs betrekking op een computersysteem; ook wanneer de informatieverwerking uitsluitend door mensen plaatsvindt is deze levenscyclus te onderkennen. Omdat computersystemen minder flexibel zijn dan mensen springt deze meer in het oog. Een gangbare waarde voor de levenscyclus ligt in de ordegrootte van 6 tot 12 jaar.

3.6. Projectbeheersingsmethoden
Om de levenscyclus beter beheersbaar te maken zijn een aantal projectbeheersingsmethoden ontwikkeld. Hierbij wordt deze levenscyclus onderverdeeld in een aantal fasen. In elk van deze fasen moet een aantal activiteiten worden uitgevoerd. Bekende projectbeheersingsmethoden zijn:
PARAET – Projekt Aanpak RAET
PRISM – Professional Information Systems Management
PROMPT – Project Resource Organization Management Planning Technique
SDM – Systems Development Methodology
Omdat deze laatste methode door vele bedrijven wordt gebruikt wordt in deze bijdrage de fase-indeling van SDM gehanteerd.

SDM werd in 1970 door Pandata ontwikkeld. In 1985 werd SDM2 geïntroduceerd, waarin de fasen op een andere wijze zijn afgebakend.
Bij SDM2 worden de volgende fasen onderkend:
Fase 0 – Informatieplanning.
Fase 1 – Definitiestudie.
Fase 2 – Basisontwerp.
Fase 3 – Detailontwerp.
Fase 4 – Realisatie.
Fase 5 – Invoering.
Fase 6 – Gebruik en beheer.

Elke fase bestaat uit een aantal activiteiten. Het resultaat van de

activiteiten wordt vastgelegd in rapporten, de zogenaamde mijlpaalprodukten. Deze mijlpaalprodukten vormen de basis voor de officiële besluitvorming; op basis hiervan wordt besloten of verder wordt gegaan op de ingeslagen weg.
SDM geeft aan wat er in elke fase moet worden vastgelegd; over hoe het moet worden vastgelegd (welke ontwerp-techniek wordt gebruikt) wordt door SDM geen uitspraak gedaan. Op de verschillende technieken die hiervoor ten dienste staan wordt in deze bijdrage ook niet verder ingegaan.

Fase 0. Informatieplanning
Tijdens de fase informatieplanning wordt een informatieplan opgesteld dat de basis is voor de vormgeving en ontwikkeling van de informatievoorziening op korte en lange termijn.
In deze fase worden geen systemen ontwikkeld. Uit de informatieplanning resulteren een tweetal rapporten: het rapport situatieanalyse en het rapport informatieplanning.
De eerste activiteit is het opstellen van een plan van aanpak. Vervolgens worden gegevens over de organisatie verzameld en wordt de veranderingsbehoefte geïnventariseerd. De huidige situatie wordt geanaliseerd en het probleem gedefinieerd. Aan de hand van deze gegevens wordt de gewenste situatie geëvalueerd en worden interessegebieden geselecteerd. De resultaten van de huidige situatie en de definitie die daaruit ontstaan is voor de taken van het informatieplan worden vastgelegd in het „Rapport situatie-analyse". Binnen de interessegebieden worden criteria vastgesteld voor de toekomstige informatievoorziening. Aan de hand hiervan wordt de informatie-architectuur ontworpen en wordt aangegeven welke delen hiervan voor automatisering in aanmerking komen. Ter verificatie worden de informatie-architectuur en de voorgestelde projecten geëvalueerd. De resultaten worden vastgelegd in het rapport „Informatieplanning", dat dient als basis voor definitiestudies.

Fase 1. Definitiestudie
De eerste activiteit bij het uitvoeren van een definitiestudie is het vastleggen van de uitgangspunten en het opstellen van een plan van aanpak. Vervolgens worden gegevens verzameld over de gewenste informatievoorziening. Deze worden vergeleken met de huidige informatievoorziening en daaruit volgen de gewenste veranderingen. Analyse van de gewenste veranderingen levert de eisen waaraan het systeem moet voldoen. Deze worden vastgelegd in een systeemconcept. Om dit systeemconcept te realiseren bestaan meerdere oplossingen. Voor de verschillende oplossingen worden de benodigde

hulpmiddelen bepaald en wordt aangegeven wat de gevolgen zijn van elk van deze oplossingen. Om uit de mogelijke oplossingen de voorkeursoplossing te bepalen worden de selectiecriteria opgesteld en wordt met deze criteria de voorkeursoplossing bepaald.

Fase 2. Basisontwerp
De eerste activiteit van deze fase is het vastleggen van de uitgangspunten en het opstellen van een plan van aanpak.
Het basisontwerp valt in twee delen uiteen: het functionele en het technische.
In de definitiestudie is een functioneel systeemconcept geschetst. Dit concept wordt verder gedetailleerd. Het is daartoe noodzakelijk om te weten in welke organisatorische omgeving het systeem gaat functioneren. Aan de hand daarvan kan worden bepaald welke gegevens door het systeem beheerd gaan worden en welke functies gebruik gaan maken van deze gegevens.
In het basisontwerp worden de functionele testplannen opgesteld. Deze vallen uiteen in een „Black-box"-testplan en een „Grey- box"-testplan.
In het „Black box"-testplan wordt gekeken vanuit het standpunt van de gebruiker en wordt het systeem dus beschouwd als een ondoordringbaar geheel waarin geen structuur te onderkennen is.
In het „Grey box"-testplan is er wel structuur te onderkennen in het systeem en wordt het systeem gezien als samenstel van deelsystemen. Dit testplan geeft aan welke functies moeten worden toegevoegd om de deelsystemen te kunnen testen.
Voor het technische deel is het relevant te weten in welke fysieke omgeving het systeem gaat draaien: op eigen computers of in een reeds bestaand systeem. In beide gevallen wordt de technische structuur waarin het systeem gaat passen vastgelegd.
Tijdens het basisontwerp kan blijken dat het systeem te omvangrijk is om als geheel te ontwikkelen. Redenen hiervoor kunnen zijn dat de ontwikkeling te lang gaat duren, dat te veel mensen moeten worden ingezet waardoor de beheersbaarheid van het project eronder lijdt of dat er niet voldoende materiedeskundigen beschikbaar zijn.
In het basisontwerp worden de deelsystemen onderscheiden die afzonderlijk te realiseren zijn.

Fase 3. Detailontwerp
De fase detailontwerp begint met het opstellen van een plan van aanpak voor deze fase. In dit plan van aanpak wordt beschreven

welke activiteiten moeten worden uitgevoerd om een gedetailleerd ontwerp te krijgen van het systeem of systeemonderdeel. In een planning wordt aangegeven wanneer en door wie de activiteiten worden uitgevoerd en wanneer afstemming plaatsvindt over de mijlpaalprodukten. Verder worden de randvoorwaarden om de activiteiten uit te voeren vastgelegd: betrokken materie-deskundigen, benodigde resources etc.
Voor het optimaal functioneren van het systeem in de organisatie is het noodzakelijk dat organisatie en systeem op elkaar afgestemd worden. In het „Rapport toekomstige organisatie" wordt vastgelegd welke verschillen er zijn tussen de huidige en de gewenste organisatie. Het beschrijft hoe en onder welke voorwaarden de benodigde veranderingen in de organisatie doorgevoerd kunnen worden.
Het „Rapport functioneel ontwerp" detailleert het betrokken systeemdeel en beschrijft de koppelingen die er tussen het systeemdeel en de omgeving zijn. Dit zijn niet alleen de koppelingen tussen de systeemdelen maar ook de dialoog met de gebruiker.
Wanneer het rapport in concept gereed is vindt een controle plaats of het voldoet aan de eisen en standaards die in de organisatie voor het functioneel ontwerp gelden.
Op welke wijze het systeem wordt gerealiseerd is beschreven in het „Rapport technisch ontwerp". Hierin wordt het systeem zover uitgewerkt dat het gereed is om geprogrammeerd te worden. Het systeem wordt beschreven van architectuur tot op modulenniveau. De interfaces met de buitenwereld en binnen het systeem zijn eenduidig gespecificeerd. Ook de database is tot op het laagste niveau uitgewerkt. Vervolgens wordt dit rapport getoetst aan de standaards en kwaliteitsnormen die in de organisatie gelden voor het technisch ontwerp.
Het „Black box"-testplan, waarvoor tijdens het basisontwerp een aanzet was gedaan wordt in het „Testplan functionele specificaties" zover gedetailleerd dat de verantwoordelijkheden en de werkwijze bij het testen duidelijk zijn en ook welke gevallen zullen worden getest.
In het „Plan voor de realisatie en invoering" wordt voor de komende twee fasen aangegeven hoeveel inspanning nodig is om het project te voltooien, welke doorlooptijd hiermee gemoeid is en welke hulpmiddelen hiervoor vereist zijn.
Als de voorgaande mijlpaalprodukten van deze fase zijn goedgekeurd, wordt ter afsluiting van de detailontwerpfase in het „Rapport detailontwerp" gerapporteerd over de stand van zaken, de kosten en de doorlooptijd van het project.

Fase 4. Realisatie
Het doel van deze fase is om het systeem dat functioneel en technisch beschreven is in het detailontwerp tot werkelijkheid te maken. In deze fase vindt de vertaling plaats van technisch ontwerp naar programmeertaal.
Of de realisatie is uitgevoerd conform het functionele ontwerp wordt getoetst tijdens de acceptatietest.
In de detailontwerpfase is het „Black box"-testplan opgesteld en goedgekeurd. Daarom kan de acceptatietest plaatsvinden aan de hand van dit testplan. Tijdens de uitvoering van de acceptatietest kan blijken dat er toch nog wensen zijn. Als de resultaten de inbedrijfstelling van het systeem niet in de weg staan, is het verstandig het systeem toch in te voeren. Hieraan zitten twee aspecten: enerzijds kunnen er meer wensen ontstaan wanneer het systeem operationeel is, anderzijds kunnen wensen die tijdens de acceptatietest geuit zijn achteraf blijken minder zwaar te wegen. Aan de hand hiervan kan een afweging worden gemaakt welke wijzigingen wel en niet zinvol zijn om door te voeren.

Fase 5. Invoering
In deze fase wordt de organisatie, waar het systeem voor bedoeld is, voorbereid op de ingebruikname van het systeem. Vervolgens wordt het systeem daadwerkelijk in gebruik genomen.
In een vroeg stadium moeten alle mensen die met het systeem te maken krijgen worden voorgelicht over de komst van het systeem. Het is belangrijk hen in een vroeg stadium te betrekken bij het opstarten. Bij de voorbereidingen voor de invoering nemen de opleidingen een belangrijke plaats in. Opleiding moet worden gegeven aan gebruikers, onderhoudsgroep en produktiepersoneel.
Wanneer het systeem een bestaand systeem gaat vervangen, dan moeten de gegevens uit het bestaande systeem geconverteerd worden.

Fase 6. Gebruik en beheer
In deze fase wordt het systeem gebruikt voor het doel waarvoor het ontwikkeld is. De taak van de beheerorganisatie is om te zorgen dat het systeem operationeel blijft. Dit gebeurt door snel herstel van geconstateerde gebreken en het aanpassen van het systeem aan veranderende omstandigheden in het proces en in de gebruikersorganisatie.
De kosten van deze fase overtreffen de totale kosten van alle eerdere fasen. In de literatuur wordt voor deze fase wel gezegd dat tweederde van de totale kosten van het systeem tijdens zijn levenscyclus wordt uitgegeven tijdens de fase gebruik en beheer.

4. Beïnvloedingsfactoren

Begroten is het bepalen van de bouwinspanning (bijv. in mensmaanden) en de doorlooptijd aan de hand van de projectomvang. De projectomvang kan bijvoorbeeld worden uitgedrukt in aantal programmaregels of aantal functiepunten. Dit wordt nader toegelicht in paragraaf 5.
Naast de omvang van het project zijn er nog vele factoren die inspanning en doorlooptijd beïnvloeden. Een aantal belangrijke beïnvloedingsfactoren worden hieronder in willekeurige volgorde genoemd.
Sommige beïnvloedingsfactoren overlappen elkaar; een systeem dat een beperking heeft ten aanzien van beschikbaar geheugen zal in het algemeen compacte en dus complexe software bevatten en valt hierdoor onder zowel de beïnvloedingsfactor „complexiteit software" als „beperking werkgeheugen".

Betrouwbaarheid software
De vereiste betrouwbaarheid van de software is afhankelijk van de toepassing; het spreekt voor zich dat aan het veiligheidssysteem voor een kerncentrale hogere eisen gesteld worden dan aan een computerspel. Hogere eisen aan een systeem brengen hogere kosten met zich mee; bijvoorbeeld meerdere gelijksoortige computers waarvan bij uitval van een van de computers de taken door een andere worden overgenomen. Betrouwbaardere systemen vragen ook om een andere en een uitgebreidere testmethode, bijvoorbeeld simulatie van het proces door een apart systeem in plaats van het testen met de doelinstallatie op trial-and-error basis.

Omvang database
Een aantal begrotingsmodellen gebruiken de omvang van de database als beïnvloedingsfactor. Sommige modellen gaan uit van het aantal verschillende entiteitstypen terwijl andere modellen het aantal occurrences als maat nemen.

Complexiteit software
Complexiteit van de software is een beïnvloedingsfactor waar moeilijk een objectieve maatstaf voor te vinden is. In de literatuur worden verschillende mogelijkheden genoemd: hoeveelheid interacties tussen de verschillende componenten van de software, de structuur van de software, de mate van nieuwheid, type applicatie. Hier komt nog bij dat de complexiteit van de software eerst in een laat stadium zichtbaar wordt namelijk in de detail-ontwerpfase. In voorliggende

fasen zijn de wensen van de klant nog te onvolledig en onduidelijk gespecificeerd om een definitieve uitspraak te kunnen doen over de complexiteit.

Beperkingen executietijd, responsietijd en werkgeheugen
Wanneer er beperkingen worden opgelegd aan de tijd die beschikbaar is voor een berekening of aan de omvang van de programma's dan moeten de programma's geoptimaliseerd worden naar snelheid of omvang. Dit optimaliseren vergroot de benodigde bouwinspanning. Door bij de keuze van de hardware rekening te houden met de gewenste snelheid en de verwachte omvang van de software kunnen optimalisatieslagen worden vermeden. In z'n algemeenheid kan worden gezegd dat het goedkoper is snelheidswinst te behalen door reeds bij aanvang snellere (= duurdere) hardware aan te schaffen dan door de software te optimaliseren. Buiten het kostenaspect is een ander nadeel van geoptimaliseerde software dat reeds bij geringe wijzigingen de optimalisatie teniet kan worden gedaan. Bovendien is geoptimaliseerde software minder doorzichtig en daardoor moeilijker onderhoudbaar.

Ervaring met soortgelijke applicaties
Bij de evaluatie van projecten komt vaak naar voren dat sommige delen van het project anders waren gerealiseerd als de huidige ervaring al bij het begin van het project aanwezig was geweest. Wanneer het bouwteam ervaring heeft met soortgelijke applicaties dan zullen een aantal van deze „valkuilen" worden vermeden.

Kwaliteit analisten en programmeurs
Goede analisten vertalen de klant-eisen naar consistente ontwerpen die realiseerbaar zijn voor de programmeurs.
Goede programmeurs combineren zorgvuldigheid met een hoge produktiviteit. Zorgvuldigheid uit zich in goed gedocumenteerde en goed geteste programma's. Goed gedocumenteerde programma's zijn belangrijk voor een onderhoudbaar systeem.

Ervaring met apparatuur en programmeertaal
Ervaring met apparatuur betekent dat kennis aanwezig is van de mogelijkheden en onmogelijkheden van de gebruikte apparatuur. Hetzelfde geldt voor ervaring met programmeertaal.

Gebruik moderne programmeertechnieken
De moderne programmeertechnieken zijn erop gericht de produktiviteit te verhogen en het systeem beter onderhoudbaar te maken. Dit

wordt gerealiseerd door een overzichtelijke structuur in de programma's aan te brengen en door dubbelingen in het systeem te voorkomen. De overzichtelijke structuur wordt bereikt door de programma's op te bouwen uit bouwstenen die klein zijn (typisch 1 á 2 bladzijden). Dubbelingen worden voorkomen door de programma bouwstenen die algemeen bruikbaar zijn in een bibliotheek op te nemen en te verplichten deze waar mogelijk toe te passen.

Gebruik van hulpmiddelen
Er bestaat een breed spectrum aan hulpmiddelen dat zich uitstrekt van ondersteuning bij foutzoeken tot gereedschappen voor informatie-analyse en -ontwerp. Al deze hulpmiddelen hebben tot doel de produktiviteit te verhogen en fouten in een vroeg stadium te detecteren. Ook het gebruik van de vierde generatie programmeertalen verhoogt de produktiviteit. „Vierde" duidt op het aantal generaties van programmeertalen die zijn ontwikkeld sinds de uitvinding van de computer.

Eisen aan projectduur
De optimale projectduur is naar twee zijden begrensd. Verkorten van de doorlooptijd van het project betekent een groter bouwteam waardoor de inspanning voor coördinatie toeneemt. Bij een te lange doorlooptijd is het moeilijk het team gemotiveerd te houden. Grote projecten kunnen worden opgesplitst in deelprojecten, die apart worden ingevoerd en elk afzonderlijk wel een acceptabele doorlooptijd hebben.

Hergebruik software en gebruik van softwarepakketten
Het hergebruik van software kan op allerlei niveaus voorkomen: een systeemconcept, een deelsysteem tot modules waarmee bijvoorbeeld conversie van de ene eenheid naar de andere eenheid wordt uitgevoerd. In welke mate hergebruik van software plaatsvindt wordt mede bepaald door de wijze waarop dit wordt gestimuleerd: als er een uniforme werkwijze is voor de projecten, als er een beschrijving is van gerealiseerde projecten of een software-bibliotheek waarin veel voorkomende modules en (voorbeeld) programma's zijn opgenomen dan zal hergebruik van software frequenter voorkomen dan wanneer dit gebeurt op basis van informele contacten en „horen zeggen".
Buiten het hergebruik van specialistische software bestaan er ook softwarepakketten voor algemene toepassingen. Voorbeelden hiervan zijn database-pakketten, schermbesturingspakketten etc.

Samenstelling bouwteam
De produktiviteit van het bouwteam is niet de som van de individuele kwaliteiten van de leden van het bouwteam. Als een bouwteam meerdere leiderfiguren bevat dan kan dit leiden tot rivaliteit, hetgeen de produktiviteit vermindert. Een teveel aan creatieve probleemoplossers kan ertoe leiden dat zij trachten elkaar in vindingrijkheid te overtroeven, eveneens met bovengenoemd resultaat.
Verder wordt de gewenste teamsamenstelling ook bepaald door de projectfase: in de ontwerpfase ligt de nadruk op creativiteit, terwijl tijdens de realisatiefase zorgvuldige programmeurs belangrijk zijn.

Mate van detail en consistentie uitgangsdocument
De mate van detail van het uitgangsdocument wordt bepaald door de fase van het project waar het document bij hoort (definitiestudie, basisontwerp etc). Verder moet het over de gehele linie eenzelfde mate van detail hebben en het moet consistent zijn.

Aard van het systeem (real-time of batch)
Zoals eerder aangegeven kunnen systemen worden ingedeeld naar de wijze en snelheid waarop op veranderingen in de omgeving wordt gereageerd (real-time- en batch-systemen). Hierbij zijn de real-time-systemen complexer dan batch-systemen omdat op alle mogelijke verstoringen uit de omgeving gereageerd moet kunnen worden.

5. Begrotingsmodellen

Door onderzoekers zijn de gegevens van een groot aantal projecten bestudeerd en is getracht een relatie te vinden tussen de kenmerken van die projecten en de hoeveelheid inspanning en doorlooptijd die met die projecten gemoeid was. Dit heeft geleid tot een aantal begrotingsmodellen. Deze zijn in drie hoofdgroepen onder te verdelen:
– parametrische modellen;
– het analoge model;
– overige modellen.

Bij de parametrische modellen worden bouwinspanning en doorlooptijd bepaald door middel van formules. Een aantal bekende parametrische modellen worden beschreven in paragraaf 5.2.
Bij het analoge model wordt de bouwinspanning bepaald door vergelijking met soortgelijke, eerder gerealiseerde projecten.

Naast de genoemde modellen zijn er ook een aantal modellen die wel in een geautomatiseerde versie beschikbaar zijn maar waarvan de interne werking niet openbaar is gemaakt. Voorbeelden hiervan zijn Estimacs (ontwikkeld door H. Rubin van het Hunter College) en PRICE (van Radio Corporation of America).
Op deze „geheime" modellen wordt verder niet ingegaan.
Daarnaast zijn er nog een tweetal begrotingsmethoden waaraan andere gegevens dan de projectomvang ten grondslag liggen en daarom niet worden aanbevolen. Dit zijn:
– de „Price-to-win"-methode;
– de capaciteitmethode.

De „Price-to-win"-methode baseert zich op de prijs waarvoor men verwacht het project nog binnen te kunnen halen („We verwachten dat de concurrentie dit project voor 2 miljoen heeft aangeboden en we moeten zeker 10 procent hieronder blijven"). Hiermee wordt aangenomen dat de concurrentie wel een begroting heeft gemaakt en hiermee aan de ruime kant is gebleven. Deze aanname is moeilijk te controleren. Daarom is bovengenoemde redenering als begrotingsmethode riskant en niet aan te bevelen. Wanneer strategische motieven aan deze keuze ten grondslag liggen („Een voet tussen de deur te krijgen") is het verstandig het project door middel van een model te begroten en dan te besluiten of de winst-/verliesmarge acceptabel is.
De capaciteitsmethode gaat ervan uit dat het project gerealiseerd moet worden met de mens-capaciteit die vrijgemaakt kan worden. Een gevaar dat hierbij dreigt ligt voor de hand: deze capaciteit kan blijken te klein te zijn met als gevolg budget- en levertijdoverschrijding. Anderzijds bestaat ook het risico dat de Wet van Parkinson („Work expands to fill the available volume") in werking treedt. Er worden dan allerlei ongevraagde faciliteiten ontwikkeld.

5.1. Werkwijze voor het opstellen van een begroting

Heemstra [2] onderkent bij het opstellen van een begroting voor een softwareproject globaal de volgende stappen:
1. kalibreer het model;
2. decomponeer het project;
3. bepaal omvang van de software;
4. vertaal omvang naar benodigde inspanning;
5. corrigeer aan de hand van kostenbeïnvloedende factoren;
6. verdeel de inspanning over de projectfasen;
7. bepaal de doorlooptijd;
8. voer gevoeligheids- en risico-analyse uit.

Niet alle modellen voeren de stappen in de onderstaande volgorde uit. Dit wordt bij de beschrijving van het betreffende model aangegeven.

Stap 1. Kalibreer het model
Elk bedrijf heeft een eigen werkwijze voor het ontwikkelen van software. Dit betekent dat ook de formules die gebruikt worden bij het bepalen van bouwinspanning en doorlooptijd af zullen wijken van de formules die door andere organisaties worden gebruikt. Om het model geschikt te maken voor toepassing in de organisatie moet het model worden gekalibreerd, dat wil zeggen op een bepaalde groep van toepassingen worden toegesneden.
Het is daartoe noodzakelijk om te beschikken over de gegevens van een groot aantal soortgelijke projecten die in dezelfde omgeving zijn gerealiseerd.
Indien deze gegevens niet aanwezig zijn (bijv. bij een eerste project) zal bij deskundigen te rade moeten worden gegaan.

Stap 2. Decomponeer het project
Een voorwaarde om een begroting van een project te kunnen maken is duidelijkheid over de omvang van het project: welke functionaliteit moet door het systeem geleverd worden, wie stelt het protocol op voor de acceptatietest, wie vervaardigt de bijbehorende testgegevens, als hiervoor simulatieprogrammatuur noodzakelijk is wie zorgt dan daarvoor, wie leidt de gebruikers en de systeembeheerder op etc. Gedetailleerd inzicht in de functionaliteit van het project kan verkregen worden door functionele decompositie toe te passen. Welke activiteiten in het kader van het project worden uitgevoerd kan worden bepaald door dit te toetsen aan de hand van alle activiteiten die door bijvoorbeeld SDM2 in de verschillende fasen worden onderkend.

Stap 3. Bepaal omvang van de software
Afhankelijk van het gehanteerde model wordt de omvang van het project uitgedrukt in regels code, functiepunten (FPA) of mensuren (analoge model).

Stap 4. Vertaal omvang naar benodigde inspanning
De wijze waarop deze stap wordt uitgevoerd hangt af van het gevolgde model.

Stap 5. Corrigeer aan de hand van kostenbeïnvloedingsfactoren
In de vorige stap is een „netto" inspanning bepaald. Deze moet nog worden gecorrigeerd met de beïnvloedingsfactoren uit paragraaf 4. De totale beïnvloedingsfactor wordt gevonden door de gekozen waarden van de afzonderlijke beïnvloedingsfactoren met elkaar te vermenigvuldigen. De totale inspanning is de „netto" inspanning maal de totale beïnvloedingsfactor.
Welke waarden toepasselijk zijn voor de beïnvloedingsfactoren is tijdens stap 1 bepaald aan de hand van historische projectgegevens.

Stap 6. Verdeel de inspanning over de projectfasen
De totale inspanning wordt verdeeld over de projectfasen. Welke verhouding hangt af van de definitie die de gevolgde projectfasering voor de verschillende projectfasen geeft. Ook hiervoor wordt gebruik gemaakt van historische projectgegevens.

Stap 7. Bepaal de doorlooptijd
Uit de totale inspanning kan een doorlooptijd worden bepaald. De doorlooptijd is zowel naar beneden als naar boven toe begrensd: een kortere doorlooptijd vereist een onevenredig grote inspanning aan coördinatie; een langere doorlooptijd geeft problemen om de teamleden gemotiveerd te houden.
De wijze waarop de doorlooptijd wordt bepaald hangt af van het gekozen begrotingsmodel.

Stap 8. Voer gevoeligheids- en risico-analyse uit
Door de beïnvloedingsfactoren met kleine stapjes te variëren en het effect hiervan na te gaan op inspanning en doorlooptijd wordt een indruk gekregen hoe gevoelig de begroting is voor wijzigingen en misschattingen van de beïnvloedingsfactoren.
Het doel van risico-analyse ([4] en [5]) is het onderkennen van de risico's die de doelstelling van het project (levering van de gevraagde functionaliteit binnen budget en planning) in de weg kunnen staan zodat maatregelen kunnen worden getroffen om de risico's te minimaliseren. Het fundament voor de methode is gelegd op Harvard Business School en is in Zweden verder ontwikkeld door het „Riksdataforbundet" onder de naam „SarBachets Analys", in het Engels vertaald tot „Security By Analysis". De methode bestaat uit een lijst van ruim 180 vragen, die de volgende aandachtsgebieden bestrijken:
1. project-afbakening;
2. methodologie en standaards;
3. project-procedures;
4. project-organisatie;

5. rapportage en controle;
6. technische infrastructuur;
7. bemanning;
8. belangrijke eigenschappen van het project;
9. interne vragen.

Enkele aandachtsgebieden zijn weer verder onderverdeeld in deel-aandachtsgebieden.
Op elke vraag zijn meerdere antwoorden mogelijk en aan elk antwoord is een risicowaarde gekoppeld dat varieert tussen 0 (= geen risico) en een maximum dat afhangt van het gewicht van de vraag. Deze risicowaarden worden per (deel-)aandachtsgebied en over alle vragen getotaliseerd. Het totaal voor een gebied gedeeld door de maximaal haalbare risicowaarde voor dat gebied levert het risicopercentage. Een hoog risicopercentage voor een gebied of voor enkele vragen uit een gebied duidt erop dat maatregelen gewenst zijn.

5.2. Parametrische modellen

De aantal bekende parametrische modellen zijn:
– COCOMO (COnstructive COst MOdel);
– PRICE (Programmed Review of Information Costing Evaluation);
– PUTNAM;
– FPA (Funktie Punt Analyse);
– Walston en Felix.

5.2.1. COCOMO

Het COCOMO-model is in 1981 ontwikkeld door Boehm [1] en is zeer uitgebreid beschreven. Van COCOMO bestaan drie modellen. Deze modellen variëren van eenvoudig (Basic COCOMO) tot zeer gedetailleerd (Detailed COCOMO). Tussen deze modellen bevindt zich het Intermediate COCOMO. Alleen het Basic COCOMO wordt beschreven; Intermediate en Detailed COCOMO zijn een verdere uitwerking van Basic COCOMO en verschillen voornamelijk in het aantal beïnvloedingsfactoren waarmee rekening wordt gehouden. In het COCOMO wordt rekening gehouden met de omgeving waarin de ontwikkeling plaatsvindt; hierbij wordt onderscheid gemaakt tussen:
– Organic mode: kleine teams, geringe tijdsdruk.
– Semi-detached mode: tussen organic en embedded in.
– Embedded mode: veel beperkingen met betrekking tot de

ontwikkelomgeving bijvoorbeeld tijdsdruk, beperkte beschikbare bouwcapaciteit, complexe combinatie van hard- en software.

Bij het ontwikkelen van het COCOMO is gebruik gemaakt van de projecthistorie van 63 projecten van verschillende aard (wetenschappelijk, administratief, procescontrole) die in de periode 1964 tot 1979 zijn gerealiseerd. Op basis van deze projectgegevens ontwikkelde Boehm de onderstaande formules voor inspanning en doorlooptijd, rekening houdend met de omgeving waarin de ontwikkeling plaatsvindt.
De coëfficiënten in tabel 1 zijn ter indicatie; omdat de ontwikkelomgeving, de werkwijze en de aard van de projecten afwijkt van de projecten die Boehm gebruikte bij het ontwikkelen van zijn methode is het wenselijk deze voor gebruik in de eigen organisatie te ijken.

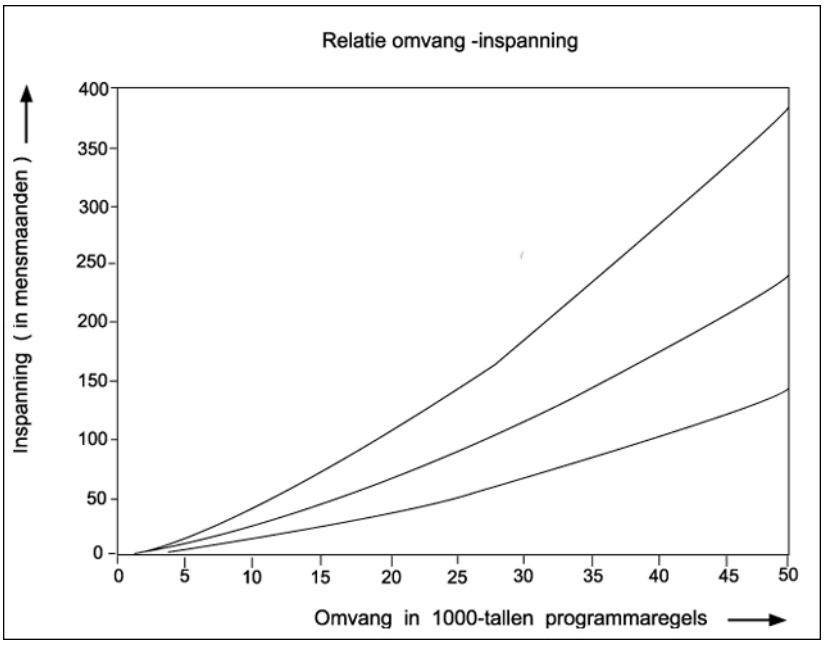
Organic:	Inspanning	$= 2{,}4 * \text{Omvang}^{1{,}05}$
	Doorlooptijd	$= 2{,}5 * \text{Inspanning}^{0{,}38}$
Semi-detached:	Inspanning	$= 3{,}0 * \text{Omvang}^{1{,}12}$
	Doorlooptijd	$= 2{,}5 * \text{Inspanning}^{0{,}35}$
Embedded:	Inspanning	$= 3{,}6 * \text{Omvang}^{1{,}20}$
	Doorlooptijd	$= 2{,}5 * \text{Inspanning}^{0{,}32}$

Hierin is: Inspanning uitgedrukt in mensmaanden.
Doorlooptijd uitgedrukt in maanden.
Omvang uitgedrukt in duizendtallen programmaregels.

Tabel 1.

De relatie tussen omvang en inspanning wordt weergegeven in figuur 2. Hierin geldt de bovenste grafieklijn voor de Embedded mode, de middelste voor de Semi-detached mode en de onderste voor de Organic mode.
Bij deze formules horen een aantal randvoorwaarden:
Met programmaregels worden die regels bedoeld die naar machinetaal worden omgezet; commentaarregels worden niet meegerekend.
De met de formules berekende inspanning en doorlooptijd heeft betrekking op de fasen van basisontwerp tot en met de acceptatietest aan het einde van de realisatie met uitzondering van het vervaardigen en testen van opleidingen. Verder is alleen rekening gehouden met de activiteiten die meegenomen zijn in de decompositie.

0838-0542



Figuur 2

Er wordt alleen rekening gehouden met projectactiviteiten; activiteiten van ondersteunende afdelingen (systeembeheer, secretaresses, topmanagement) zijn niet in de uitkomsten begrepen.
Er wordt vanuit gegaan dat het project goed gemanaged wordt, dat wil zeggen dat leegloop wordt geminimaliseerd.
Als basis voor de begroting moet een stabiele specificatie dienen die niet meer aan grote wijzigingen onderhevig zijn.
COCOMO houdt rekening met een vijftiental beïnvloedingsfactoren, hier cost-drivers genoemd. In tabel 2 zijn de analyses van de genoemde projecten en de mening van een aantal experts verwerkt.

Beïnvloedingsfactor	Waarde factor					
	erg laag	laag	gem.	hoog	erg hoog	extra hoog
Betrouwbaarheid software	0,75	0,88	1,00	1,15	1,40	
Omvang database		0,94	1,00	1,08	1,16	
Complexiteit software	0,70	0,85	1,00	1,15	1,30	1,65
Beperkingen executietijd			1,00	1,11	1,30	1,66
Beperkingen werkgeheugen			1,00	1,06	1,21	1,56
Vervangingsgraad computer		0,87	1,00	1,15	1,30	
Responsietijd		0,87	1,00	1,07	1,15	
Kwaliteit analisten	1,46	1,19	1,00	0,86	0,71	
Ervaring met soortgelijke applicaties	1,29	1,13	1,00	0,91	0,82	
Kwaliteit programmeurs	1,42	1,17	1,00	0,86	0,70	
Ervaring met apparatuur	1,21	1,10	1,00	0,90		
Ervaring met programmeertaal	1,14	1,07	1,00	0,95		
Gebruik moderne programmeer-technieken	1,24	1,10	1,00	0,91	0,82	
Gebruik software tools	1,24	1,10	1,00	0,91	0,83	
Eisen aan projectduur	1,23	1,08	1,00	1,04	1,10	

Tabel 2.

5.2.2. PUTNAM

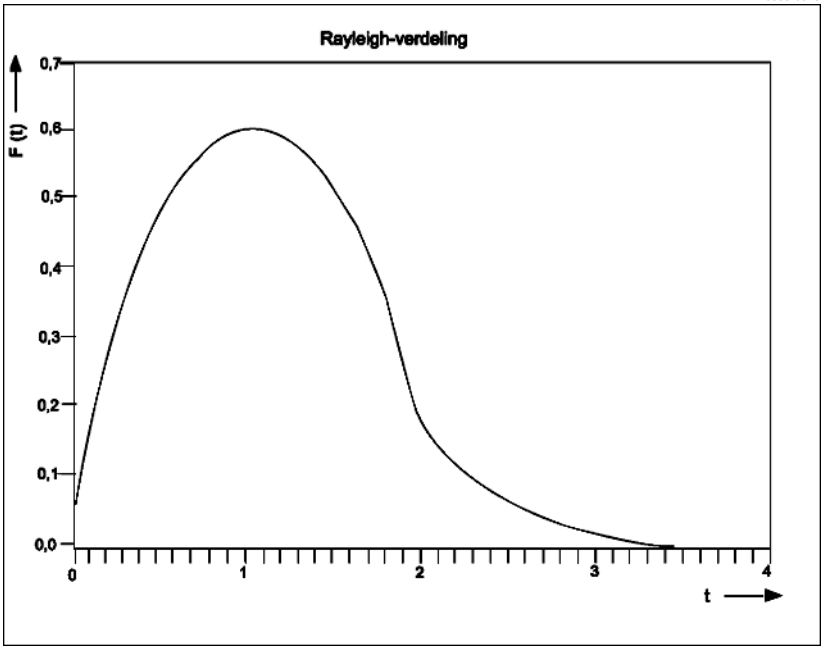
Het model van Putnam werd ontwikkeld in 1974 en is gebaseerd op een onderzoek van Norden. Deze laatstgenoemde maakte frequentieverdelingen van het aantal mensen dat tijdens de looptijd van een project werd ingezet en ontdekte dat deze verdelingen grote overeenkomst vertoonden met de Rayleigh-verdeling. Er werden geen redenen gevonden voor deze overeenkomst, het was als het ware een „natuurwet".
De Rayleigh-verdeling voldoet aan de vergelijking:

$$F(t) = t \ast e^{\frac{-t^2}{2}}$$

Hierin is:
t = de tijd
e = grondtal van de natuurlijke logaritme

Deze verdeling is hierna in figuur 3 grafisch weergegeven.

0838-0543



Figuur 3.

Putnam bouwde voort op het onderzoek van Norden en ging er daarbij vanuit dat:

1. de totale oppervlakte van de functie overeenkomt met de totale inspanning die tijdens de volledige levenscyclus van een systeem benodigd is;
2. het systeem wordt opgeleverd op het moment dat de functie zijn maximale waarde (en dus de benodigde menskracht maximaal is), dat wil zeggen op tijdstip t = 1.

Het oppervlak tussen t = 0 en t = 1 (de bouwinspanning) is 39,45 procent van het totale oppervlak van de functie. Dus 60,55 procent van de inspanning wordt verricht tijdens de beheer- en onderhoudsfase. Deze cijfers kloppen met ervaringscijfers en ondersteunen de aannamen van Putnam.
Na wiskundige uitwerking kwam Putnam tot onderstaande formules voor inspanning en doorlooptijd:

Inspanning = a ✷ Omvang^{b}
Doorlooptijd = c ✷ Inspanning^{d}

Hierin hebben de variabelen a, b, c en d waarden die afhangen van de aard van het project, de werkwijze in de organisatie en de samenstelling van het bouwteam.
Putnam komt dus tot soortgelijke formules als Boehm bij het COCOMO-model, zij het langs een andere weg.
Putnam gebruikte vijftig NASA-projecten om waarden voor a, b, c en d te bepalen. Omdat deze voor elke organisatie afzonderlijk moeten worden geijkt aan de hand van de projectgegevens van gerealiseerde projecten is er van afgezien deze hier te vermelden.

5.2.3. FPA

FPA (Funktie Punt Analyse) is in de periode van 1974-1979 door dr. A. Albrecht van IBM ontwikkeld als gereedschap voor het meten van produktiviteit en is later onder meer door dr. E. Rudolph (University of Auckland, New Zealand) verfijnd tot begrotingsmodel.
FPA wordt uitgevoerd aan de hand van een functionele beschrijving van het systeem en gaat er vanuit dat aantal en complexiteit van gebruikersfuncties en genormaliseerde gegevensgroepen een maat zijn voor de omvang van het systeem.
De functies worden verdeeld in invoerfuncties, uitvoerfuncties, opvragingsfuncties en koppelingsfuncties.
Zowel functies als gegevensgroepen worden in drie moeilijkheidsklassen ingedeeld: eenvoudig, gemiddeld en moeilijk.
Afhankelijk van de moeilijkheidsklasse wordt aan elke functie en gegevensgroep een aantal punten toegekend, de zogenaamde functiepunten.
De som van deze functiepunten geeft de omvang van het project in bruto functiepunten.
De bruto functiepunten worden gecorrigeerd met de beïnvloedingsfactoren en leveren dan de netto functiepunten. Het aantal netto functiepunten maal het aantal uren per functiepunt resulteert in de verwachte inspanning.
Hieronder volgt een stapsgewijze beschrijving van deze procedure.

Stap 1. Kalibratie van het model

Het model wordt gekalibreerd door van een aantal gelijksoortige gerealiseerde projecten een begroting uit te voeren met behulp van FPA. Het aantal uren per functiepunt verschilt van organisatie tot organisatie en is onder andere afhankelijk van de automatiseringsgraad in de gebruikersorganisatie, de generatie van de gebruikte programmeertaal en de omvang van het project. Het aantal uren per functiepunt dat in de literatuur genoemd wordt varieert van 1 uur

per functiepunt (normale systeemontwikkeling, 4e generatietaal, kleine projecten) tot 20-50 uur per functiepunt (omgeving met lage automatiseringsgraad, 3e generatietaal, grote projecten).
De grote spreiding in uren per functiepunt benadrukt de noodzaak om het model voor gebruik te kalibreren.

Stap 2. Decompositie van het project
Het is voor FPA voldoende over een functionele beschrijving van het systeem te beschikken, bijvoorbeeld definitiestudie, basisontwerp of detailontwerp. Hierin moet voldoende duidelijk beschreven staan welke interfaces het systeem heeft naar de gebruiker en naar aangrenzende systemen en moet het datamodel beschreven zijn.

Stap 3. Bepaal omvang van het project
FPA gaat er vanuit dat de voor de gebruiker zichtbare functies en gegevensgroepen maatgevend zijn voor de omvang van het project. Voor de functies wordt onderscheid gemaakt tussen invoer-, uitvoer-, opvragings- en koppelingsfuncties. Hieronder wordt aangegeven wat de omschrijving is van deze functies en van een logische gegevensgroep.
In z'n algemeenheid geldt dat functies die meerdere malen voorkomen maar een volledig identieke verwerking hebben, slechts een keer worden geteld.

Invoerfunctie
Invoerfuncties zijn alle functies die gegevens ontvangen van buiten het systeem met het doel deze, al dan niet na bewerking, vast te leggen in de database. Deze gegevens kunnen afkomstig zijn van bijvoorbeeld een toetsenbord, van een ander systeem via datacommunicatie, sensoren etc. Vaak is de invoerfunctie gecombineerd met een opvraagfunctie: eerst wordt informatie opgevraagd om deze vervolgens te wijzigen. In dat geval is sprake van zowel een invoer- als een opvraagfunctie (zie tabel 3).

		Aantal invoervelden		
		1-4	5-15	>15
Geraadpleegde	0-1 E	E	G	
Logische	2	E	G	M
Gegevensgroepen	>2	G	M	M

Tabel 3.

Uitvoerfunctie
Uitvoerfuncties zijn functies die gegevens uit de database, al dan niet na bewerking, naar de buitenwereld voeren. Als de gegevens op aanvraag naar buiten worden gevoerd en dit gebeurt op basis van een eenduidig zoekargument dan wordt dit niet geteld als een uitvoerfunctie, doch als een opvraagfunctie. Uitvoerfuncties die dezelfde informatie leveren maar op verschillende wijze gerangschikt tellen als een uitvoerfunctie.
Achtergrondinformatie die niet afkomstig is uit de database (vaste tekst, tijd, systeemidentificatie etc.) telt niet mee bij de bepaling van het aantal uitvoervelden (zie tabel 4).

		Aantal uitvoervelden		
		1-5	6-19	>19
Geraadpleegde	0-1 E	E	G	
Logische	2-3	E	G	M
Gegevensgroepen	>3	G	M	M

Tabel 4.

Opvragingsfunctie
Een opvragingsfunctie is een uitvoerfunctie die door zoekargumenten zodanig beperkt is dat de opvraging tot een eenduidig antwoord leidt. De bepaling van de moeilijkheidsgraad valt daarom in twee delen uiteen.
1. de moeilijkheidsgraad van de ingevoerde zoekargumenten (zie tabel 5);
2. de moeilijkheidsgraad van het uitgevoerde resultaat (zie tabel 6).

Voor beide delen wordt de moeilijkheidsgraad bepaald, de moeilijkheidsgraad van de opvraagfunctie is gelijk aan de moeilijkheidsgraad van de meest complexe van de twee.

		Aantal invoervelden		
		1-4	5-15	>15
Geraadpleegde	0-1 E	E	G	
Logische	2	E	G	M
Gegevensgroepen	>2	G	M	M

Tabel 5. Moeilijkheidsgraad zoekargumenten.

		Aantal uitvoervelden		
		1-5	6-19	>19
Geraadpleegde	0-1 E	E	G	
Logische	2-3	E	G	M
Gegevensgroepen	>3	G	M	M

Tabel 6. Moeilijkheidsgraad resultaat.

Koppelingsfunctie
Een koppelingsfunctie komt voor wanneer het te ontwikkelen systeem gekoppeld moet worden aan bestaande systemen die de gegevens in een vorm aanbieden of willen ontvangen die niet aansluit bij de database (zie tabel 7). Er moet dan een conversie plaatsvinden van de gegevens. Als de koppeling zodanig is dat er geen conversie hoeft plaats te vinden dan is sprake van een in- of uitvoerfunctie.

		Aantal attributen		
		1-19	20-50	>50
Geraadpleegde	0-1 E	E	G	
Logische	2-5	E	G	M
Gegevensgroepen	>5	G	M	M

Tabel 7.

Logische gegevensgroepen
De logische gegevensgroep moet genormaliseerd zijn tot in de derde normaalvorm (zie tabel 8).

		Aantal attributen		
		1-19	20-50	>50
Logische Gegevensgroepen	1	E	E	G

Tabel 8.

Bepaling aantal functiepunten
Als voor alle gebruikersfuncties en gegevensgroepen de moeilijkheidsklasse is bepaald kan in tabel 9 het aantal functiepunten dat bij de functie of gegevensgroep hoort worden opgezocht.

	Moeilijkheidsklasse		
	E	G	M
Invoerfunctie	3 4	6	
Uitvoerfunctie	4	5	7
Opvragingsfunctie	3	4	6
Koppelingsfunctie	5	7	10
Logische gegevensgroep	7	10	15

Tabel 9.

Het bruto aantal functiepunten voor het systeem is de som van de functiepunten voor alle functies en gegevensgroepen.

Stap 4. Correctie aan de hand van beïnvloedingsfactoren
In de vorige stap is het bruto aantal functiepunten bepaald. Het netto aantal functiepunten wordt bepaald uit de formule:

Netto functiepunten = bruto functiepunten ✷ correctiefactor

In deze correctiefactor wordt rekening gehouden met de hierna vermelde veertien beïnvloedingsfactoren.

1. gebruik van datacommunicatie binnen het systeem;
2. gebruik van gedistribueerde gegevensverwerking binnen het systeem;
3. eisen aan response-tijd;
4. eisen aan geheugengebruik en processing-time;
5. aantal transacties;
6. gebruik van on-line data entry;
7. conversationele in- en uitvoer;
8. bestanden/database wordt on-line bijgewerkt;
9. complexiteit van het informatiesysteem;
10. mate van hergebruik;
11. complexiteit conversie en ingebruikname;
12. bedieningsgemak;
13. geïnstalleerd op meerdere plaatsen;
14. aanpasbaarheid van het systeem.

Aan elk van deze factoren wordt een waarde toegekend van 0 tot en met 5. Hierbij is:
0 = niet van toepassing/geen invloed;
1 = af en toe van invloed;
2 = matige invloed;
3 = normale invloed;

4 = belangrijke invloed;
5 = zwaarwegende invloed.

De invloed van de beïnvloedingsfactoren is bij FPA beperkt tot plus of min 35 procent van het aantal bruto functiepunten.
De totale waarde van alle beïnvloedingsfactoren kan waarden aannemen tussen 0 (14 ✷ 0) en 70 (14 ✷ 5). De correctiefactor is daarom:

Correctiefactor = 0,65 + (totale beïnvloeding) / 100

Stap 5. Vertaal omvang naar benodigde inspanning
Het aantal netto functiepunten wordt omgerekend naar inspanning door het aantal netto functiepunten te vermenigvuldigen met het aantal uren per functiepunt zoals dit tijdens de kalibratie voor een project met deze omvang is bepaald.

Stap 6. Bepaling doorlooptijd
FPA geeft geen oplossing voor het bepalen van de doorlooptijd. Dit kan worden uitgevoerd zoals beschreven bij het analoge model.

5.3. Analoge model
Bij het analoge model wordt er ook gebruik gemaakt van de gegevens van analoge gerealiseerde (deel-)projecten; in dit geval zijn deze niet samengevat in coëfficiënten of uren per functiepunt maar wordt rechtstreeks gebruik gemaakt van het aantal uren dat bij de gerealiseerde projecten is besteed aan de ontwikkeling van bepaalde subsystemen.
Een variant van de analoge methode is de zogenaamde „Expert”-methode; het project wordt hierbij door een of meer deskundigen begroot. De ervaringscijfers zijn hierbij niet expliciet vastgelegd maar worden vervangen door de ervaring van de deskundige(n). Een kanttekening die hierbij gemaakt wordt is dat het gemiddelde automatiseringsproject 1 à 2 jaar duurt; een expert met 10 jaar ervaring baseert zijn begroting dus op circa 7 projecten; een gering aantal vergeleken met het aantal dat mogelijk is wanneer de projecthistorie expliciet wordt vastgelegd.

Stap 1. Kalibreer het model
Kalibratie van het model vindt plaats door gegevens te verzamelen van zoveel mogelijk gelijksoortige gerealiseerde projecten. Het is hierbij noodzakelijk het aantal bestede uren gedetailleerd vast te

leggen, of dit netto- of bruto-uren zijn en de geschatte waarden voor de beïnvloedingsfactoren.

Stap 2. Decomponeer het project
De mate van functionele decompositie is afhankelijk van de fase waarin het zich bevindt.

Stap 4. Vertaal omvang naar benodigde inspanning
Deze stap valt in twee delen uiteen:
– het bepalen van de bouwinspanning;
– het bepalen van een toeslag voor overhead-activiteiten.

a. Bepaal bouwinspanning
Als het uitgangsdocument een definitiestudie, een basisontwerp of een functioneel ontwerp is dan is hierin alleen de gewenste functionaliteit vastgelegd en niet de wijze waarop dit gerealiseerd gaat worden. Tijdens het technisch ontwerp wordt vastgelegd op welke wijze het project gerealiseerd gaat worden en dan worden een aantal componenten toegevoegd die in de eerder genoemde functionele beschrijvingen van het systeem nog niet zichtbaar waren. Hierbij moet worden gedacht aan communicatie met de gebruiker via beeldschermen, communicatie met aangrenzende systemen, alarmering van de gebruiker bij onverwachte situaties etc. Daarnaast moet mogelijk nog testsoftware ontwikkeld worden voor het testen van de afzonderlijke subsystemen. Met deze toegevoegde componenten moet rekening worden gehouden wanneer een schatting wordt gemaakt van de omvang van het project.
De som van de uren die nodig is voor hiervoorstaande activiteiten is de netto inspanning.

b. Bepaal toeslag voor overhead-activiteiten
Niet alle tijd die ten laste komt aan het projectbudget wordt direct besteed aan het ontwikkelen van mijlpaal-produkten; een deel van de tijd wordt gebruikt voor overhead-activiteiten: teamvergaderingen, werkbesprekingen, bijdragen aan kwaliteitscontroles.
Een reële waarde hiervoor is ongeveer een halve dag per week, dat wil zeggen circa 10 procent van de tijd die ten laste komt van het projectbudget wordt besteed aan overhead-activiteiten.
Verder wordt ook tijd besteed aan coördinatie en rapportage door projectleider en eventueel teamleiders. Een richtlijn hiervoor is 1 dag per week per medewerker dat wil zeggen een team van 5 mensen heeft een projectleider nodig die zich 5 dagen per week bezig houdt met coördinatie.

De totale inspanning wordt bepaald door de bouwinspanning te vermenigvuldigen met 1,1 ✷ 1,2 = circa 1,3.

Stap 5. Corrigeer aan de hand van kostenbepalende factoren
Uit de projecthistorie van de referentieprojecten blijkt onder welke omstandigheden de bouw is uitgevoerd: welke programmeertaal, door ervaren mensen, welke methodiek en techniek is gehanteerd, hergebruik van software etc. Als deze bij het te begroten project afwijkt moet hiermee rekening worden gehouden.

Stap 6. Verdeel de inspanning over de projectfasen
Voor de verdeling van de inspanning over de projectfasen wordt gebruik gemaakt van ervaringscijfers. Ter illustratie: in de literatuur wordt voor de verhouding basisontwerp: detailontwerp: realisatie de getallen 2 : 10 : 5 genoemd.

Stap 7. Bepaal de doorlooptijd
Om te bepalen wat de doorlooptijd wordt moet het aantal uren bekend zijn dat jaarlijks aan projectactiviteiten wordt besteed. Dit is per bedrijf verschillend, de hierna vermelde berekening geeft een voorbeeld.

Aantal dagen per jaar			365	
Af: Weekends (52 ✷ 2)		104		
Feestdagen	9			
Cursussen	15			
Ziekte	12			
Vakantie	25			
Arbeidsduurverkorting (ADV)	15			
Bedrijfsactiviteiten	10			
	+ —			
Werkdagen niet ten laste van project		86		
		+ —		
Totaal niet-project dagen			190	
			– —	
Projectdagen			175	✷ 8 = 1400 uur/jaar

Een project dat een totale inspanning vraagt van 10.000 mensuren en een maximale doorlooptijd van 2 jaar vereist dus een team dat bestaat uit 10.000 mensuren/(2 jaar ✷ 1400 uur/jaar) = 3,6 mensen. Dit wordt afgerond op 4 mensen, de meerwaarde aan bouwcapaciteit wordt besteed om het team geleidelijk te laten groeien en aan het einde af te laten nemen.

Er zijn praktische grenzen aan doorlooptijd en teamgrootte. Voor doorlooptijd ligt deze grens bij gelijkblijvende bezetting op circa 2 jaar, voor de teamgrootte op 8 à 9 mensen. Wanneer een project groter is dan ruwweg 18 mensjaren is het verstandig te overwegen of het mogelijk is het project in deelprojecten op te splitsen.

6. Literatuur

[1] Boehm, Barry W., *„Software engineering economics”*, Prentice Hall, 1981, ISBN 0-13-822122-7.
[2] Heemstra, F. J., *„Het begroten van software projecten: Meten is Weten”*, Eindhoven University of Technology, Eindhoven 1987, ISBN 90-6757-028-1.
[3] Heemstra, Fred J., *„Hoe duur is programmatuur”*, Kluwer Bedrijfswetenschappen, Deventer, 1989, ISBN 90-267-1371-1.
[4] Rijsenbrij, D. B. B. en A. H. Bauer, „Projektdiagnose: goed begin is het halve werk”, *Informatie* jaargang 31, nr. 3.
[5] Rowold, Paul, *„Schatten & begroten van software-projecten”*, Tutein Nolthenius, Amsterdam, 1989, ISBN 90-72194-08-X.
[6] Turner, W. S., R. P. Langerhorst, H. B. Eilers, G. F. Hice, R. J. M. M. Castermans, L. F. Cashwell en A. A. Uijttenbroek (editor), *„System development methodology”*, (samenvatting, voorlopige versie), Pandata, Rijswijk, 1985.
[7] Turner, W. S., R. P. Langerhorst, G. F. Hice, H. B. Eilers en A. A. Uijttenbroek, *„System development methodology”* (Nederlandstalige versie), Pandata, Rijswijk, 1989, ISBN 90-71996-09-3.