

## 6 Softwarekwaliteit, wat is dat?

### 6.1 Inleiding

Software wordt in steeds meer en in een steeds grotere diversiteit van toepassingsgebieden gebruikt. De juiste werking is vaak van wezenlijk belang voor het succes van een bedrijfsvoering en ook voor de veiligheid van mensen. Het goed specificeren en evalueren van softwarekwaliteit is noodzakelijk om kwaliteit te kunnen realiseren en te kunnen garanderen. In deel I zijn verschillende definities van kwaliteit van Garvin behandeld. In dit tweede deel van het boek wordt met name uitgegaan van Garvin's 'User Based' en 'Product Based' definities van kwaliteit, ook wel de gebruikersgerichte en de productgerichte benadering genoemd. We maken daarbij de volgende kanttekeningen:

- Het begrip 'User Based' komt ter sprake wanneer het begrip kwaliteitsbehoefte en het van daaruit afleiden van kwaliteitseisen voor software wordt behandeld.
- 'Product based' staat centraal wanneer softwarekwaliteit wordt opgedeeld in een aantal kwaliteitskenmerken.
- Verder kan worden gesteld dat ook Garvin's definitie 'Value Based' een rol speelt ofwel de waardegerichte benadering van kwaliteit, met name wanneer wordt gesproken over het stellen van prioriteiten voor te specificeren en te realiseren kwaliteitskenmerken.

Dit hoofdstuk heeft de volgende structuur. Paragraaf 6.2 gaat in op kwaliteitskenmerken van software. Kwaliteitskenmerken zijn van belang voor ontwikkelaars om door middel van een goed ontwerpproces kwaliteit aan software te kunnen meegeven. Anderzijds zijn kwaliteitskenmerken van belang voor softwaregebruikers om te kunnen aangeven waar voor hen softwarekwaliteit door wordt bepaald. Goede definities van kwaliteitskenmerken zijn nodig om kwaliteit bespreekbaar en hanteerbaar te maken. Paragraaf 6.3 beschrijft de 'state of the art' van het definiëren van softwarekwaliteit aan de hand van de ISO/IEC-standaard. In paragraaf 6.4 wordt beschreven dat afhankelijk van de situatie software op verschillende wijze kan worden beschouwd, en afhankelijk daarvan ook kwaliteitskenmerken kunnen variëren. Paragraaf 6.5 sluit het hoofdstuk af met conclusies.

### 6.2 Softwarekwaliteit, verschillende opvattingen

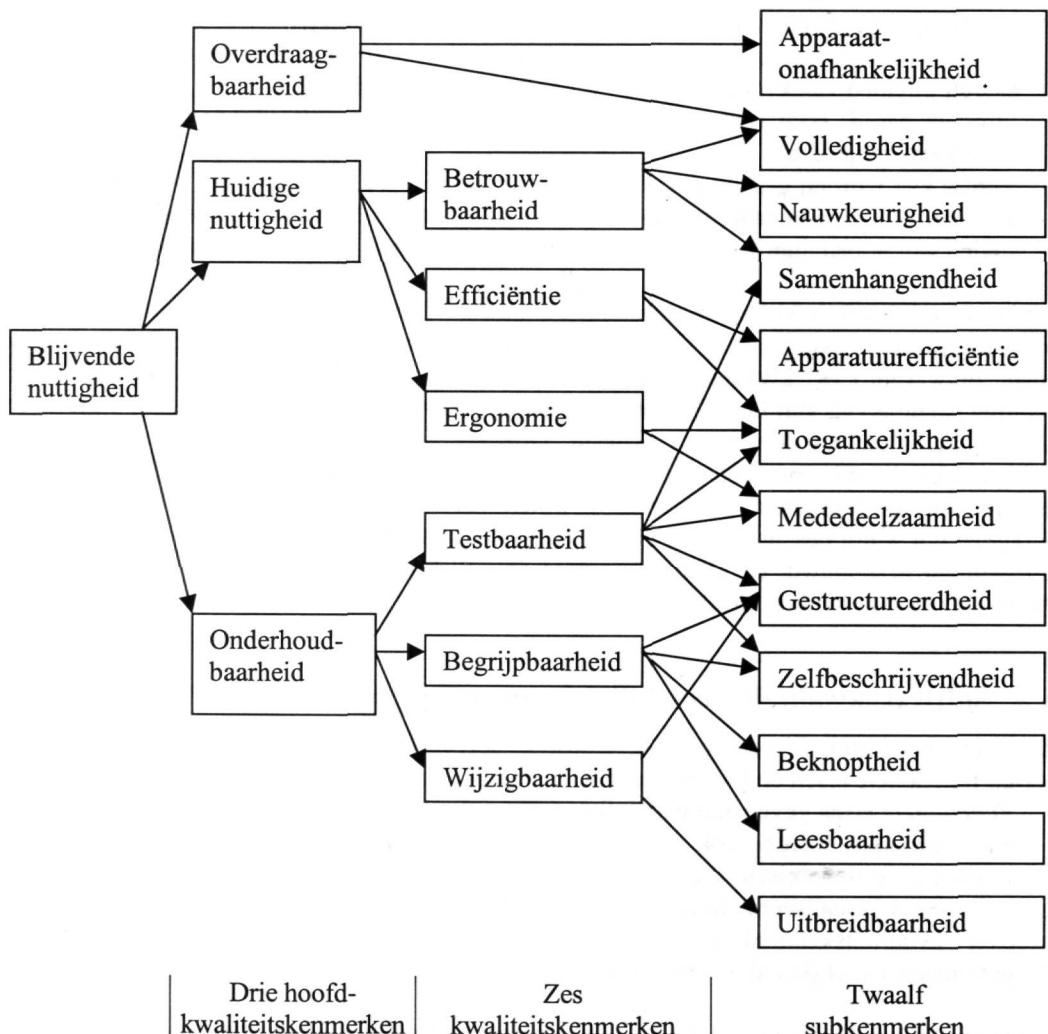
Zoals in hoofdstuk 3 werd gesteld, wordt in softwareontwikkeling onderscheid gemaakt tussen functionele eisen en kwaliteitseisen.

- *Functionele eisen* geven aan **wat** software moet doen.
- *Kwaliteitseisen* worden ook wel aangeduid met de term niet-functionele eisen. Bedoeld worden specifieke eisen, die betrekking hebben op de manier waarop functionaliteit van software is gerealiseerd (**hoe**). Wanneer voldaan is aan kwaliteitseisen beschikt software over kwaliteitskenmerken. Voorbeelden van kwaliteitseisen zijn betrouwbaarheid, gebruiksvriendelijkheid, responsesnelheid, en onderhoudbaarheid.

Boehm (1976) en Cavano en McCall (1978) kunnen in de software-industrie worden beschouwd als vertegenwoordigers van 'het eerste uur' als het gaat om het beschrijven van kwaliteitskenmerken. In hoofdstuk 3 werd reeds daaraan gerefereerd. Softwarekwaliteit wordt door deze auteurs beschreven aan de hand van een goed gedefinieerde verzameling kwaliteitskenmerken.

Verschillende andere onderzoekers hebben in de afgelopen decennia voortgebouwd op de eerste inzichten. Genoemd kunnen worden Willmer (1985), Deutsch en Willis (1987) en Bemelmans (1998). We zullen in het navolgende de kwaliteitsopvattingen van deze auteurs bespreken waarbij we in eerste instantie nog niet ingaan op de precieze betekenis van de afzonderlijke kwaliteitskenmerken en hun subkenmerken. Bij de besprekking van de verschillende opvattingen zullen we steeds weer de volgende drie invalshoeken hanteren:

- *wat staat centraal in de kwaliteitsopvatting?* We bespreken hier kort de verzameling kwaliteitskenmerken, de omvang en de structuur van die verzameling.
- *wie bepaalt kwaliteit?* Aan de orde komen de partijen die binnen de kwaliteitsopvatting worden verondersteld gebruik te maken van de kwaliteitskenmerken.
- *hoe wordt kwaliteit gerealiseerd?* Kort wordt aandacht besteed aan de manier waarop kan worden omgegaan met de kwaliteitskenmerken om kwaliteit in software ‘in te bouwen’.



Figuur 6.1: Kwaliteit volgens Boehm et al (1976)

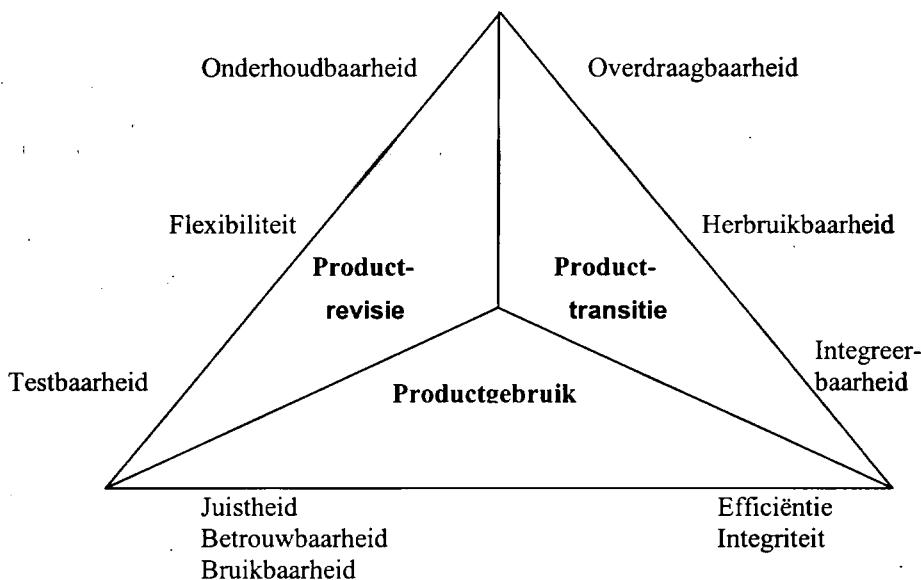
### 6.2.1 Kwaliteit volgens Boehm en Cavano en McCall (1976)

Wellicht het meest bekend op het gebied van kwaliteit in de software-industrie is de ‘boom’-structuur van kwaliteitskenmerken van Boehm, zie figuur 6.1.

*Wat staat centraal in de kwaliteitsopvattingen van Boehm, Cavano en McCall?*

De genoemde structuur voor softwarekwaliteit staat bekend als de ‘boom van Boehm’ maar is in feite een netwerkstructuur (de vorm van figuur 6.1 geeft dat al aan). Het is de eerste poging in de literatuur om het begrip softwarekwaliteit hanteerbaar te maken. De ‘boom’ bestaat uit drie hoofdkwaliteitskenmerken, zes kwaliteitskenmerken en een groot aantal subkenmerken. Uitgangspunt is het begrip ‘blijvende nuttigheid’ dat is onderverdeeld in de drie hoofdkenmerken ‘overdraagbaarheid’, ofwel de verplaatsbaarheid van software, ‘huidige nuttigheid’, ofwel geschiktheid voor gebruik, en onderhoudbaarheid. Uit deze drie hoofdkenmerken van kwaliteit worden vervolgens langs hiërarchische weg kwaliteitskenmerken, zoals bijvoorbeeld betrouwbaarheid, gebruiksvriendelijkheid en begrijpbaarheid afgeleid en hieruit weer subkenmerken van kwaliteit, zoals bijvoorbeeld nauwkeurigheid en leesbaarheid.

Cavano en McCall kijken iets anders tegen kwaliteit aan. Zij maken onderscheid in kwaliteitskenmerken vanuit drie invalshoeken, te weten ‘product operations’ ofwel de werking van software, ‘product revision’ ofwel het aanpassen van software, en ‘product transition’ ofwel het (laten) evolueren van software naar een volgende generatie. Binnen elk van deze invalshoeken worden vervolgens kwaliteitskenmerken gedefinieerd, zie figuur 6.2.



Figuur 6.2: Kwaliteit volgens Cavano en McCall (1978)

Cavano en McCall maken voor het onderscheiden van kwaliteitskenmerken gebruik van de levenscyclus van software. Afhankelijk van de fase waarin software zich bevindt, of die software doorloopt, zijn verschillende kwaliteitskenmerken relevant. We merken op dat Cavano en McCall naast de invalshoeken voor softwarekwaliteit en de kwaliteitskenmerken ook een taal

hebben geformuleerd die zo goed als mogelijk begrijpelijk is voor gebruikers. Dit in tegenstelling tot de kwaliteitskenmerken in de 'boom van Boehm' die meer in ontwerperstaal zijn beschreven.

### *Wie bepaalt kwaliteit in de kwaliteitsopvattingen van Boehm, Calvano en McCall?*

Op grond van de overwegend technische terminologie die wordt gehanteerd, kan worden gesteld dat de beide besproken benaderingen zich richten zich op het ontwerp ofwel de (technische) realisatie van kwaliteitskenmerken in software. De ontwerper wordt voorzien van een eenduidig begrippenkader in zijn streven naar kwalitatief goede software.

### *Hoe wordt kwaliteit bewerkstelligd in de kwaliteitsopvattingen van Boehm, Calvano en McCall?*

Subkenmerken worden afgeleid uit kwaliteitskenmerken. Hoge kwaliteit lijkt afhankelijk te zijn van het aantal (sub)kenmerken dat software tijdens ontwikkeling meekrijgt. De subkenmerken, die van een sterk technische signatuur zijn, vormen het uitgangspunt voor te nemen maatregelen en een basis voor het inrichten van het ontwerp- en programmeerwerk.

#### **6.2.2 De kwaliteitsopvatting van Willmer (1985)**

Willmer (1985) gaat een stap verder met het specificeren van kwaliteitskenmerken dan voorgaande auteurs. Zij geeft een model waarin kwaliteitskenmerken, opgesteld door klanten, zijn gerelateerd aan kenmerken die ontwerpers in software kunnen aanbrengen, zie figuur 6.3.

		Kwaliteitskenmerken van klanten/afnemers					
		Productkenmerken van de ontwikkelaars / makers					
		- Taalgebruik (duidelijke naamgeving, lengte zinnen, gebruik goto, enz.)					
		- Controle mogelijkheden in de software (access control)					
		- Mate van standaardisatie					
		- Mate van structurering					
		- Documentatie					
		- Visualisering					
Begrijpbaarheid		X	X	X	X		X
Veranderbaarheid		X	X	X		X	X
Overdraagbaarheid op andere hardware			X	X		X	X
Onderhoudbaarheid		X	X	X			X
Te koppelen met andere software			X	X	X		
Mogelijkheid van hergebruik		X	X	X			
Testbaarheid		X	X	X			X
Efficiëntie van de software				X		X	X
Beveiliging (autorisatie gebruik)						X	
Correctheid (vrij van fouten)					X		
Gevoeligheid voor fouten						X	
Herstartbaarheid na foutoptreden						X	

Figuur 6.3: Kwaliteitskenmerken en productkenmerken (Willmer 1985)

Ook de kwaliteitsopvatting van Willmer wordt weer vanuit de drie invalshoeken *wat*, *wie* en *hoe* toegelicht.

#### *Wat staat centraal in de kwaliteitsopvatting van Willmer?*

De twee verzamelingen van kenmerken die door Willmer worden onderscheiden zijn enerzijds een verzameling softwarekwaliteitskenmerken vanuit het perspectief van softwaregebruik en anderzijds een verzameling softwarekenmerken vanuit ontwikkelperspectief. Doel van dit onderscheid is de communicatie tussen gebruikers van software en ontwikkelaars te verbeteren. Gebruikerseisen betreffende de kwaliteit kunnen aan de hand van een goed gedefinieerde terminologie eenduidig worden vertaald naar een verzameling ‘productkenmerken’.

#### *Wie bepaalt kwaliteit in de kwaliteitsopvatting van Willmer?*

Twee (hoofd)groepen betrokkenen worden onderscheiden, te weten de ontwikkelaars en de gebruikers of de klanten. Gebruikers dienen hun behoeften aan kwaliteit onder woorden te brengen. Ontwikkelaars nemen vervolgens op basis daarvan maatregelen om bepaalde softwarekenmerken te realiseren. Op de horizontale as in de figuur zijn een aantal ( $n$ ) productkenmerken in de taal van ontwikkelaars weergegeven (bijvoorbeeld ‘taalgebruik in de software’, ‘mate van structurering’ etc.). Op de verticale as staan een aantal ( $m$ ) kwaliteitskenmerken in de taal van de gebruikers of de klanten, bijvoorbeeld: ‘begrijpbaarheid’, ‘herstartbaarheid na foutoptreden’. Willmer’s model kan worden beschouwd als een aanzet om de twee werelden van ontwikkelaars en gebruikers naar elkaar te brengen.

#### *Hoe wordt kwaliteit bewerkstelligd in de kwaliteitsopvatting van Willmer?*

Willmer wijkt af van het opdoen (hiërarchisch) afleiden van subkenmerken uit geaggregeerde of hoofdkwaliteitskenmerken zoals dat door Boehm wordt beschreven. Door een netwerk van ( $n:m$ )-relaties te beschrijven tussen twee aparte verzamelingen wordt de vertaling van kenmerken uit de ene naar de andere verzameling ondersteund. Uiteraard spelen hierbij tal van beperkingen en randvoorwaarden een rol, die zowel van organisatorische als van technische signatuur kunnen zijn. Bijvoorbeeld de standaardisatie die een ontwerper wenst te hanteren moet vaak afgestemd worden op geldende organisatiestandaards, en mogelijkheden voor visualisering (grafische weergave-mogelijkheden, kleuren en geluid) hangen vaak samen met de mogelijkheden van de technologie waarover men beschikt.

### **6.2.3 De kwaliteitsopvatting volgens Bemelmans (1998)**

Door Bemelmans (1998) wordt onderscheid gemaakt tussen gebruikseisen, beheereisen en eisen met betrekking tot informatie (zie figuur 6.4).

#### *Wat staat centraal in de kwaliteitsopvatting van Bemelmans?*

Het onderscheid tussen gebruikseisen en beheereisen komt in grote lijnen overeen met het onderscheid tussen deelverzamelingen van kwaliteitskenmerken in de eerder genoemde opvattingen van Boehm en Cavano & McCall. Belangrijk verschil vormt echter de aparte deelverzameling van kwaliteitseisen betreffende de informatie die door software wordt gegenereerd. Gesteld wordt dat de kwaliteitseisen die aan informatie kunnen worden gesteld

vaak een uitgangspunt vormen voor het formuleren van kwaliteitseisen over het gebruik en het beheer van software.

#### *Wie bepaalt kwaliteit in de kwaliteitsopvatting van Bemelmans?*

De kwaliteitseisen met betrekking tot informatie zijn sterk bedrijfscontext- of organisatiegebonden. Deze eisen komen over het algemeen op intersubjectieve wijze tot stand, dat wil zeggen door gebruikers in gezamenlijk overleg. Verder wordt in deze kwaliteitsopvatting, in tegenstelling tot de voorgaande opvattingen, benadrukt dat beheer ook eisen stelt aan softwarekwaliteit. In plaats van het ontwerpperspectief, dat in de twee voorgaande kwaliteitsopvattingen werd genoemd, wordt in Bemelmans' opvatting het veel bredere gebruik- en beheerperspectief gehanteerd.

#### *Hoe wordt kwaliteit bewerkstelligd in de kwaliteitsopvatting van Bemelmans?*

Voor wat betreft het realiseren van softwarekwaliteit komt deze opvatting in grote lijnen overeen met de twee voorgaande opvattingen, namelijk kwaliteitseisen dienen te worden vertaald naar bepaalde kenmerken. Bemelmans richt zich met name op het 'voortraject' ofwel het afleiden van kwaliteitseisen uit de karakteristieken van een organisatie. Gesteld wordt dat kwaliteitseisen in tegenstelling tot functionele eisen in mindere mate kunnen worden afgeleid van de te ondersteunen operationele processen, taken of procedures. Veel beter kan worden uitgegaan van de kenmerken van het type organisatie, en bijvoorbeeld de manier waarop processen worden aangestuurd (Bemelmans 1986). In lijn met deze opvatting worden later in dit deel van het boek benaderingen voor het specificeren van kwaliteitseisen beschreven (zie hoofdstuk 6).

Eisen voor het product INFORMATIE	Eisen voor het SYSTEEM	
	Eisen voor het GEBRUIK	Eisen voor het BEHEER
- Juistheid - Volledigheid - Actualiteit - Controleerbaarheid - Nauwkeurigheid - aggregatie - selectie	- Tijdigheid - Integriteit - juistheid - volledigheid - betrouwbaarheid - Security / beveiliging - Doelmatigheid - Effectiviteit - Gebruiksvriendelijkheid	- Flexibiliteit - Onderhoudbaarheid - Testbaarheid - Portabiliteit - Inpasbaarheid in de technische infrastructuur - Herbruikbaarheid

Figuur 6.4: Kwaliteitseisen vanuit drie invalshoeken Bemelmans (1998)

## **6.2.4 Conclusies**

Het bovenstaande laat zien dat verschillende auteurs in de afgelopen decennia hebben gestreefd naar een zo goed mogelijke beschrijving van softwarekwaliteit. De overeenkomsten en verschillen van de verschillende opvattingen hebben de roep om een eenduidig en operationeel begrippenstelsel voor softwarekwaliteit versterkt. Alleen standaardisatie lijkt tegemoet te kunnen komen aan die behoefte. In het navolgende bespreken we dan ook het werk dat de afgelopen jaren werd verricht op het terrein van standaardisatie van softwarekwaliteit binnen de internationale standaardisatiegemeenschap ISO/IEC.

## **6.3 Softwarekwaliteit: op weg naar een standaard (ISO/IEC 9126)**

In de internationale standaardisatiegemeenschap zijn de afgelopen jaren verdere stappen gezet op het terrein van het specificeren, het realiseren en het evalueren van softwarekwaliteit. Binnen de ISO/IEC 9126 standaard wordt een onderscheid gemaakt tussen interne softwarekwaliteit, externe softwarekwaliteit en softwarekwaliteit-in-gebruik. De standaard wil voor het omgaan met softwarekwaliteit een consistente terminologie bieden en is bedoeld voor uiteenlopende groepen praktijkmensen, zoals betrokken bij selectie en inkoop van software, ontwikkeling van software, het specificeren van kwaliteitseisen, het gebruiken van de software, de evaluatie van software, en onderhoudsdeskundigen. Voorbeelden van activiteiten waarbij gebruik kan worden gemaakt van de ISO/IEC 9126 standaard zijn:

- het identificeren van kwaliteitseisen
- het valideren van een definitie van eisen
- het bepalen van software-ontwerpdoelstellingen
- het identificeren van testdoelstellingen
- het vaststellen van criteria voor kwaliteitsborging
- het bepalen van de acceptatiecriteria van eenmaal gerealiseerde software.

Binnen ISO/IEC9126 worden twee delen onderscheiden, respectievelijk

- een deel waarin de begrippen interne en externe kwaliteit worden behandeld en
- een deel waarin kwaliteit-in-gebruik nader wordt beschreven.

Met name met het recentelijk ontwikkelde begrip kwaliteit-in-gebruik wordt getracht nadrukkelijker aan te sluiten bij de dagelijkse praktijk van een klant of softwaregebruiker.

### **6.3.1 Interne en externe kwaliteit**

Het eerste deel van de ISO-standaard behandelt de begrippen interne en externe kwaliteit en de relaties daartussen. Met interne en externe kwaliteit worden in feite het ontwikkelperspectief en het gebruiksperspectief op softwarekwaliteit bedoeld. Interne kwaliteit is de kwaliteit die door een ontwikkelaar gerealiseerd kan worden door bepaalde maatregelen te nemen tijdens het ontwikkelproces en door bepaalde kenmerken in software in te bouwen. Externe kwaliteit is de kwaliteit zoals die door gebruikers of klanten wordt gewenst, geëist of ervaren. Interne en externe kwaliteit zijn eigenlijk de twee zijden van de ‘medaille van softwarekwaliteit’. Voor deze softwarekwaliteit worden zes zogeheten ‘hoofd-kwaliteitskenmerken’ beschreven, te weten functionaliteit, betrouwbaarheid, gebruiksvriendelijkheid, efficiëntie, onderhoudbaarheid en portabiliteit. De definities luiden als volgt.

### *Functionaliteit*

De mate waarin software in functies voorziet die voldoen aan gestelde en impliciete behoeften, wanneer de software wordt gebruikt onder gespecificeerde omstandigheden.

### *Betrouwbaarheid*

De mate waarin software een bepaald niveau van werking en prestatie biedt, wanneer de software wordt gebruikt onder gespecificeerde omstandigheden.

### *Gebruiksvriendelijkheid*

De mate waarin software inzichtelijk is, men zich de software eigen kan maken, de software gebruikt kan worden, en aantrekkelijk is voor een gebruiker, wanneer de software wordt gebruikt onder gespecificeerde omstandigheden.

### *Efficiëntie*

De mate waarin software de juiste prestatie (performance) biedt, in relatie tot de hoeveelheid middelen die worden gebruikt, onder gespecificeerde omstandigheden.

### *Onderhoudbaarheid*

De mate waarin software kan worden gemodificeerd. Modificaties omvatten correcties, verbeteringen of aanpassingen van de software aan veranderingen in de omgeving, en in de eisen en de functionele specificaties.

### *Portabiliteit*

De mate waarin software kan worden overgebracht van de ene (hardware)omgeving naar een andere.

Elk van deze hoofdkenmerken is zorgvuldig gedefinieerd en vervolgens verder opgedeeld in subkenmerken (appendix 2).

Voor elk van de subkenmerken worden door ISO/IEC 9126 vervolgens twee verzamelingen metrieken aangereikt. Een verzameling zogeheten interne metrieken is bestemd voor de ontwikkelaar om metingen te verrichten aan de software, met name tijdens ontwikkeling. Aan de hand daarvan kan worden bepaald of de inspanningen om kwaliteit te realiseren succesvol verlopen. De verzameling externe metrieken kan door gebruikers worden gehanteerd om vast te kunnen stellen of software werkt en presteert zoals is beloofd en/of afgesproken. Externe softwarekwaliteit heeft namelijk betrekking op het gedrag van de software zoals die in de praktijk wordt gebruikt. Voor voorbeelden van interne en externe metrieken voor softwarekwaliteit wordt verwezen naar hoofdstuk 9.

Uiteraard is het de bedoeling dat de externe softwarekwaliteit het resultaat is van de interne kwaliteit zoals die is gerealiseerd door een ontwikkelaar. De relaties zijn echter uiterst complex en de ISO/IEC standaard waagt zich dan ook niet aan een analyse en beschrijving daarvan.

## **6.3.2 Kwaliteit-in-gebruik**

Het tweede deel van de standaard beschrijft het begrip kwaliteit-in-gebruik. Kwaliteit-in-gebruik is onderverdeeld in vier kenmerken, te weten effectiviteit, productiviteit, veiligheid en tevredenheid.

De definitie van kwaliteit-in-gebruik luidt:

*De mate waarin software een specifieke groep van gebruikers in een specifieke gebruiksomgeving in staat stelt om hun doelstellingen op effectieve en productieve wijze te bereiken, en op een veilige manier en naar tevredenheid.*

Ook de vier kenmerken van kwaliteit-in-gebruik zijn zorgvuldig gedefinieerd en van metrieken voorzien. We geven de definities in het navolgende. Voor voorbeelden van metrieken voor kwaliteit-in-gebruik wordt verwezen naar hoofdstuk 9.

#### *Effectiviteit*

De mate waarin software gebruikers in staat stelt om hun doelstellingen te realiseren op een nauwkeurige en volledige manier

#### *Productiviteit*

De mate waarin software gebruikers in staat stelt om de juiste hoeveelheid aan middelen aan te wenden in relatie tot de effectiviteit die in een bepaalde gebruiksomgeving dient te worden gerealiseerd.

N.B.: middelen hebben bijvoorbeeld betrekking op 'tijd om een taak te verrichten', de 'gebruikersinspanning', 'materialen' of de 'financiële kosten van gebruik'. In deel IV wordt op deze zaken nader ingegaan.

#### *Veiligheid*

De mate waarin door software wordt voorzien in acceptabele risiconiveaus voor wat betreft gevaar voor mensen, bedrijfsschade en schade aan eigendom in een specifieke gebruiksomgeving.

N.B.: risico's hebben hier met name betrekking op de mate waarin wordt afgeweken van de eerder genoemde zes hoofdkwaliteitskenmerken.

#### *Tevredenheid*

De mate waarin software in een bepaalde gebruiksomgeving tevredenheid bij gebruikers bewerkstelligt.

N.B.: tevredenheid is de reactie van gebruikers op de interactie met de software, en omvat ook de attitude of houding van gebruikers ten opzichte van het gebruik van de software. Onder gebruikers valt elk type gebruiker variërend van opdrachtgevers, gebruikersmanagement en onderhoudspersoneel.

Uiteraard dient er een relatie te bestaan tussen kwaliteit-in-gebruik, de vier kenmerken daarvan en de zes eerder genoemde hoofdkwaliteitskenmerken van softwarekwaliteit. ISO/IEC stelt dat kwaliteit-in-gebruik in feite de resultante is van de manier waarop en de mate waarin de zes hoofdkwaliteitskenmerken zijn gerealiseerd. Daarbij wordt opgemerkt dat het doel van het bewerkstelligen van softwarekwaliteit niet 'perfecte kwaliteit' is, maar noodzakelijke en voldoende kwaliteit voor de specifieke gebruiksomgeving waarvoor de software wordt ontwikkeld.

Voor wat betreft het bepalen van kwaliteit-in-gebruik, in termen van de vier kenmerken, wordt opgemerkt dat expliciet gemaakte kwaliteitsbehoeften van gebruikers niet altijd hun werkelijke behoeften weergeven. Redenen daarvan zijn onder meer dat:

- gebruikers zich niet altijd bewust zijn van hun werkelijke behoeften
- gebruikersbehoeften kunnen veranderen nadat ze zijn geformuleerd
- het praktisch gezien onmogelijk is om alle taakgebieden, werkprocessen en gebruiksmogelijkheden diepgaand te analyseren.

Ook zal het niet altijd mogelijk zijn om kwaliteitseisen voorafgaand aan het eigenlijke ontwikkelwerk volledig te specificeren in termen van de zes hoofdkwaliteitskenmerken. In dat geval dienen de specificaties van de kwaliteitseisen tijdens ontwikkeling, bijvoorbeeld op basis van prototyping, te worden bijgesteld of aangepast.

Zoals tussen kwaliteit-in-gebruik en kwaliteit vanuit het externe perspectief relaties bestaan, zo bestaan ook tussen het externe en interne perspectief van softwarekwaliteit relaties. Aan de hand van de interne en externe metrieken kan worden getracht deze onderlinge relaties aan te tonen.

ISO/IEC stelt dat proceskwaliteit, ofwel de kwaliteit van ontwikkelprocessen, bijdraagt aan het verbeteren van softwarekwaliteit, en dat de eenmaal gerealiseerde softwarekwaliteit bijdraagt aan kwaliteit-in-gebruik. Daarom is ook het evalueren en verbeteren van processen een middel om softwarekwaliteit te verbeteren, evenals dat het evalueren en verbeteren van software kwaliteit een middel is om kwaliteit-in-gebruik te verbeteren. Omgekeerd kan het evalueren van kwaliteit-in-gebruik leiden tot het aanbrengen van verbeteringen in de zes hoofdkenmerken van softwarekwaliteit en kan het evalueren van de zes hoofdkenmerken leiden tot het aanbrengen van verbeteringen in ontwikkelprocessen.

Opgemerkt dient te worden dat de kwaliteit van de informatie, die wordt gegenereerd door software, niet in de ISO 9126 standaard wordt behandeld. We geven daarom hieronder een korte toelichting op dit onderwerp

### **6.3.3 Kwaliteit van informatie**

Over de kwaliteit van informatie zijn de afgelopen jaren verschillende belangrijke artikelen verschenen. Strong e.a. (1997) laat zien dat meer en meer bedrijven de kwaliteit van informatie wensen te benoemen en te beschrijven. De gebrekige kwaliteit van data en de informatie die wordt gegenereerd door software, veroorzaakt schadeposten van miljarden dollars (Wang, 1995).

Het artikel (Strong e.a., 1997) stelt dat aandacht voor de zogeheten intrinsieke kwaliteit van data, zoals gespecificeerd en geïmplementeerd in software, niet voldoende is om gebruikers of organisaties tevreden te stellen en/of afdoende te ondersteunen. Kwaliteit van informatie (ook wel extrinsieke kwaliteit genoemd) dient te worden benaderd vanuit de omgevingsbehoeften en/of -organisatiekenmerken van de software. Daarbij gaat het bijvoorbeeld om de behoeften van gebruikers (zoals toegankelijkheid van data) en/of vanuit een organisatiestandpunt (zoals beveiliging).

In het genoemde artikel wordt, om kwaliteit van data bespreekbaar en hanteerbaar te maken, onderscheid gemaakt in kwaliteitscategorieën. De categorieën die worden onderscheiden zijn respectievelijk:

- *intrinsieke kwaliteit*, bedoeld worden accuraatheid, objectiviteit, geloofwaardigheid en reputatie van de data
- *toegankelijkheidskwaliteit*, genoemd wordt onder meer de beveiliging van de data
- *contextuele kwaliteit*, hier worden genoemd de relevantie, de toegevoegde waarde, de tijdigheid, de volledigheid en de hoeveelheid data
- *representatiekwaliteit*, genoemd worden de interpreteerbaarheid, het gemak waarmee data kunnen worden begrepen, aantrekkelijkheid en consistentie van de representatie.

Op basis van deze indelingen en begrippen zijn problemen op het vlak van informatiekwaliteit in verschillende bedrijfssituaties onderzocht. Casestudies tonen aan dat een dergelijk raamwerk voor datakwaliteit noodzakelijk is. Aangetoond werd bijvoorbeeld dat het hanteren van dit taalgebruik leidt tot het verrijken van het inzicht binnen organisaties voor wat betreft het stellen van expliciete eisen aan datakwaliteit. Ook werd duidelijk dat het hanteren van de categorieën van datakwaliteitskenmerken kan leiden tot aanvullende eisen die dienen te worden gesteld aan de software, bijvoorbeeld voor wat betreft in te bouwen beveiligingsmechanismen.

## **6.4 Verschillende beschouwingswijzen van software: verschillende kwaliteitskenmerken**

Kwaliteit-in-gebruik van software wordt bepaald door de manier waarop gebruikers de software gebruiken binnen een bepaalde bedrijf- of productomgeving.

Dit houdt in dat de kenmerken van een bedrijf- of productomgeving het belang bepalen van de kwaliteit-in-gebruik kenmerken. Bijvoorbeeld in geval binnen een bedrijfsomgeving belangrijke managementbeslissingen moeten worden genomen op strategisch niveau en daarvoor gebruik wordt gemaakt van een informatiesysteem dan zal het kwaliteit-in-gebruik-kenmerk 'effectiviteit' van groot belang zijn. Vraagt een bedrijfsomgeving om een efficiënte ondersteuning van routinematisch werk dan is met name het kwaliteit-in-gebruik-kenmerk 'productiviteit' van belang. De stelling 'verschillende bedrijfsomgevingen: verschillende kwaliteit-in-gebruik-kenmerken' kan worden uitgebreid met de stelling 'verschillende beschouwingswijzen, verschillende kwaliteitskenmerken'. In de dissertatie van Trienekens (1994) worden drie invalshoeken gehanteerd van waaruit software kan worden bezien. In het navolgende wordt met voorbeelden aangetoond dat afhankelijk van de manier waarop software wordt beschouwd, verschillende kwaliteitskenmerken al dan niet terecht worden benadrukt. De drie invalshoeken zijn respectievelijk:

- de productbegrenzing van software
- de fasen die software doorloopt
- de uniekheid van software.

Met name binnen de eerste invalshoek worden kwaliteitskenmerken van software vaak ontzettend over het hoofd gezien. We zullen daarop kort ingaan in de navolgende paragraaf.

### **6.4.1 Kwaliteitskenmerken variëren met de productbegrenzing van software**

Software is vaak niet eenduidig afgebakend en gedefinieerd. Afwisselend wordt er onder verstaan: de programmatuur (of sourcecode), een 'softwarepakket' inclusief handleidingen en procedures, maar soms wordt ook de 'output', ofwel de informatie die wordt gegenereerd

door de software, tot die software gerekend (Bemelmans 1998). Kennelijk wisselt in de praktijk de productbegrenzing van software.

We beschrijven in deze paragraaf twee verschillende productbegrenzingen van software.

1. de 'enge' begrenzing van software als programmatuur
2. de 'ruime' productbegrenzing die is uitgebreid tot en met de directe omgeving van de software.

De consequenties van het hanteren van verschillende productbegrenzingen bij het benoemen van kwaliteitskenmerken worden met enkele voorbeelden toegelicht. Opgemerkt dient te worden dat het eigenlijk principieel onjuist is dat software in de 'enge' productbegrenzing wordt ontwikkeld en wordt verkocht aan een opdrachtgever. De 'enge' productbegrenzing houdt namelijk in dat men bepaalde kwaliteitskenmerken, die zijn gerelateerd aan de gebruiksomgeving, en daarom altijd en overal in een bepaalde mate relevant zijn voor gebruikers, min of meer bewust over het hoofd ziet. We komen daarop terug in het navolgende.

#### *Software met een 'enge' productbegrenzing ofwel Software 'an sich'*

In het model voor softwarekwaliteit van Boehm c.s. (1976) heeft kwaliteit betrekking op software 'an sich', zie figuur 6.1. Boehm geeft explicet aan dat daarmee de source-code wordt bedoeld. Door Willmer (1985) wordt ook documentatie tot de software gerekend, zowel de handleidingen voor de gebruiker als voor de beheerder. Specifieke kwaliteitskenmerken daarvan worden echter niet gegeven. We bespreken kort op welke manier in deze situatie kwaliteit wordt benaderd en gehanteerd.

#### *Kwaliteit van software met een 'enge' productbegrenzing*

Software engineers trachten kwaliteit te realiseren door de softwareprogrammatuur gestructureerd te ontwerpen, de efficiency van de programmatuur te vergroten, het computergeheugengebruik te verbeteren, de toegankelijkheid van gegevensbestanden te verbeteren, etc. Doel is om door middel van het nemen van technische (ontwerp)maatregelen kwaliteitskenmerken te realiseren zoals responsesnelheid, overdraagbaarheid en koppelbaarheid. Voor de realisatie van die eisen wordt onder meer gebruik gemaakt van technische standaards. Bij deze werkzaamheden wordt *geen* rekening gehouden met specifieke kenmerken van een gebruiksomgeving waarbinnen de software moet functioneren en presteren.

#### *Software met een ruime productbegrenzing*

In de kwaliteitsopvatting van Bemelmans (1998) is software niet beperkt tot programmatuur alleen, al dan niet voorzien van documentatie. Software dient, werkend binnen een bedrijfsumgeving, te worden beschouwd als een informatiesysteem. Tevens dient de 'output', ofwel de informatie, die door de software wordt gegenereerd daarbij te worden betrokken. De productbegrenzing van software wordt daarmee aanzienlijk uitgebreid. Immers zowel de software, de integriteit van de data die wordt gegenereerd, de bijbehorende documentatie, maar ook de procedures voor de aansluiting op de taken van gebruikers vallen nu onder de definitie van software.

De kwaliteit van software met een ruime productbegrenzing is afhankelijk van meer dan alleen de kwaliteit van de programmatuur. In een praktijkonderzoek naar de waardering van gebruikers voor software wordt dit beschreven, zie o.a Bailey en Pearson (1983). kwaliteit heeft volgens Kim (1990) dan te maken met:

- de procedures voor invoer en de uitvoer van de software
- de efficiency van de gegevensverwerking in de software
- de ondersteuning die de software biedt
- de effectiviteit van de informatie die de software genereert.

We lichten kwaliteit van software met een ruime productbegrenzing in het navolgende toe met enkele voorbeelden van kwaliteitseisen.

#### *Kwaliteit van software met een ruime productbegrenzing*

Om kwaliteitseisen van software ingepast in een bedrijfsomgeving te kunnen beschrijven, dient die omgeving te worden onderzocht. Kwaliteitseisen die dan kunnen worden afgeleid zijn bijvoorbeeld 'de aansluiting van de software op de handmatige procedures' binnen een bepaalde bedrijfsfunctie of taakgebied. Of: de gewenste mate van ondersteuning van taken binnen een betreffende bedrijfsfunctie, of de beschikbaarheid naar tijd en/of plaats van software. Voor het afleiden van dit soort kwaliteitseisen is specifieke kennis van het taakgebied en van de gebruikers daarbinnen noodzakelijk. Nog belangrijker is kennis en inzicht in een bedrijfssituatie wanneer kwaliteitseisen van de informatie, die door de software wordt gegenereerd, moet worden gespecificeerd. Deze kwaliteitseisen betreffen dan bijvoorbeeld de effectiviteit van de gegenereerde informatie voor bepaalde besluitvormingsprocessen.

Bovenstaande maakt duidelijk dat softwarekwaliteit varieert met de productbegrenzing die wordt gehanteerd. We merken nogmaals op dat een 'enge' productbegrenzing van software 'an sich', ofwel software die zich louter beperkt tot de programmatuur, in feite nooit te rechvaardigen is. Daardoor wordt het omgaan met softwarekwaliteit beperkt tot een uitsluitend technisch karwei dat ver verwijderd blijft van de kwaliteit die door een gebruiksomgeving wordt gewenst of vereist.

#### **6.4.2 Kwaliteitskenmerken variëren met de levenscyclusfasen van software**

In diverse publicaties komt de kwaliteit van tussenproducten van softwareontwikkeling aan de orde, zie bijvoorbeeld Basili en Rombach (1988), Bush en Fenton (1990). Gesteld wordt onder meer dat kwaliteitsbeheersing dient te zijn gebaseerd op het per deelproduct of tussenproduct verrichten van metingen. Kwaliteitseisen en/of kenmerken kunnen sterk variëren per fase van het ontwikkelproces of per toestand waarin de software zich bevindt.

Oplevering van de software, bijvoorbeeld overdracht aan de klant of gebruiker, betekent niet per definitie het einde van het ontwikkelwerk. Door Looyen (1993) worden verschillende toestanden beschreven waarin de software zich kan bevinden. Respectievelijk wordt onderscheid gemaakt tussen de toestand:

1. vóór oplevering van software (de ontwikkeltoestand)
2. van invoering in de organisatie (ná oplevering)
3. van beheer en exploitatie
4. van wijziging.

Met betrekking tot de laatste toestand wordt opgemerkt dat in geval van wijziging, waaronder ook uitbreiding en/of verandering valt, de software (tijdelijk) weer in een ontwikkeltoestand verkeert.

We bespreken in het navolgende twee toestanden van de software, die we respectievelijk zullen beschrijven als software tijdens projectmanagement en software tijdens productmanagement (van Genuchten 1990). Met software tijdens projectmanagement bedoelen we software die projectmatig tot stand komt. Bij het afsluiten van het project wordt de software als het ware 'over de muur gegooid', hetgeen wil zeggen dat ontwikkelaar er verder in het geheel geen verantwoordelijkheid meer voor draagt. Deze toestand komt dus overeen met de zogenoemde 'ontwikkeltoestand'. Voor software tijdens productmanagement ligt dit anders. Ontwikkelaar en opdrachtgever dragen gedurende de gehele levenscyclus een gezamenlijke verantwoordelijkheid voor de software. In feite is deze situatie een optelsom van alle vier genoemde toestanden.

#### *Toestanden van software tijdens projectmanagement*

Projectmanagement richt zich op de beheersing van het ontwikkelwerk vanaf het vastleggen van eisen tot en met de constructie van software. Volgens Basili en Rombach (1988) en (Bush en Fenton (1990) kunnen op alle deel- en tussenproducten metingen worden verricht. In overleg met projectleiders en ontwikkelaars dient afhankelijk van de situatie afgesproken te worden welke kwaliteitseisen relevant zijn en welke metrieken dienen te worden gehanteerd. Voorbeelden van tussenproducten zijn: conceptuele gegevensmodellen, programmatuur, databasestructuren etc. Een voorbeeld van een kwaliteitseis is de gestructureerdheid van de software.

Doel van projectmanagement is het opleveren van software overeenkomstig de specificaties en binnen de afgesproken tijd en kosten. Zoals eerder vermeld, weten ontwikkelaars zich bij oplevering van de software ontslagen van verantwoordelijkheden. Invoering, exploitatie, onderhoud en verandering, komen geheel en al voor rekening van de gebruiker of de opdrachtgever.

#### *Kwaliteit van software tijdens projectmanagement*

De kwaliteit van software verandert tijdens het ontwikkelwerk en naar we mogen aannemen betekent zo'n verandering een verbetering. Ontwikkelaars trachten de specificaties zo goed mogelijk te vertalen in kenmerken van de software. Tussentijds vinden verificaties en validaties plaats van analyse- en ontwerpactiviteiten. Afhankelijk van de intensiteit waarmee opdrachtgevers en/of gebruikers participeren in het ontwikkelwerk wordt door hen mede de kwaliteit bepaald. Aan kwaliteitseisen zoals gebruiksvriendelijkheid, bijvoorbeeld de inzichtelijkheid van de schermlay-out, kan dan aandacht worden besteed. Omdat ontwikkelaars na oplevering geen verantwoordelijkheden meer dragen, hebben zij er geen belang bij om al teveel tijd te besteden aan zaken als de onderhoudbaarheid, de flexibiliteit of de portabiliteit van software, dus kwaliteitseisen die een rol spelen na oplevering van de software. Deze eigenschappen zullen dan ook hooguit nominaal zijn, dat wil zeggen in overeenstemming met algemeen geldende standaards zoals die bijvoorbeeld gelden voor een programmeertaal of een hardwareplatform. Ook aan kwaliteitskenmerken zoals de inleertijd van de software behoeft eigenlijk door ontwikkelaars nauwelijks of geen aandacht te worden besteed.

#### *Toestanden van software tijdens productmanagement*

Gebruikers eisen in toenemende mate dat ontwikkelaars betrokken blijven bij software in de toestanden na oplevering. Softwareontwikkeling is niet langer beperkt tot een 'ontwerpcyclus, maar strekt zich uit over de gehele levenscyclus van de software, inclusief onderhoud, wijziging en uitbreiding. Dit wordt *softwareproductmanagement* genoemd. Zoals eerder vermeld, is het belangrijkste verschil ten opzichte van projectmanagement de doorlopende

gedeelde verantwoordelijkheid van ontwikkelaars en gebruikers voor het functioneren en de kwaliteit van software, zowel tijdens het ontwerpwerk als tijdens beheer (o.a. wijziging) en exploitatie (o.a. invoering).

#### *Kwaliteit van software tijdens productmanagement*

Softwareproductmanagement omvat de toestanden van software voor en na overdracht aan de gebruiker. Dit betekent dat ontwikkelaars en gebruikers een gezamenlijk belang hebben bij het bewerkstelligen van onderhoudbaarheid, flexibiliteit en transitiemogelijkheden van de software. Softwarekwaliteit wordt voor gebruikers afhankelijk van bijvoorbeeld de geboden begeleiding bij de invoering van de software, de overeenkomsten op serviceniveau, de afspraken over nieuwe versies van de software, de integratiemogelijkheden etc. Al deze aspecten kunnen tijdens productmanagement leiden tot specifieke eisen aan de kwaliteit van de software. Een voorbeeld van een kwaliteitseis tijdens exploitatie is onderhoudbaarheid en voorbeelden tijdens transitie zijn aanpasbaarheid, converteerbaarheid, en compatibiliteit.

Softwarekwaliteit varieert met de toestanden, ofwel de fasen, die de software tijdens de levenscyclus doorloopt. Het bovenstaande maakt duidelijk dat afhankelijk van de fasen die men wenst mee te nemen in de beschouwing verschillende kwaliteitskenmerken worden benadrukt. Of dit terecht is voor wat betreft het uiteindelijke gebruik van de software hangt af van de afspraken die ontwikkelaar en opdrachtgever onderling hebben gemaakt over de verantwoordelijkheden voor bijvoorbeeld de invoering, het onderhoud en de verbetering van de software.

#### **6.4.3 Kwaliteitskenmerken variëren met de mate van uniekheid van software**

Veel ontwikkelaars en opdrachtgevers of gebruikers kunnen het zich niet langer permitteren om voor elke bedrijfssituatie specifieke ‘maatwerksoftware’ te laten ontwikkelen. Voor ontwikkelaars is dit een gevolg van de toenemende concurrentie binnen de software-industrie. Het sneller kunnen voldoen aan specifieke wensen van klanten wordt een halszaak. Aanleidingen voor gebruikers zijn enerzijds de toename in tijd en kosten die maatwerkoplossingen met zich meebrengen en anderzijds de beperkte mogelijkheden van uitbreiding en aanpassing van dit soort software. Meer en meer wint de opvatting terrein dat hergebruik van software(componenten) een oplossing is voor genoemde problemen. In deel III van dit boek wordt nader ingegaan op het vraagstuk ‘make or buy’ en de overwegingen die een rol spelen om software zelf te (laten) maken of (standaard)software te kopen. Hergebruik komt in deel IV nader aan de orde als een middel om de kwaliteit van software te beïnvloeden. Doel van hergebruik is om met een gelimiteerd aantal basiscomponenten software samen te stellen waarmee tegemoet wordt gekomen aan de wensen van een klant. Afhankelijk van de standaardcomponenten die worden gebruikt is software meer of minder uniek. Uiteraard kan samengestelde software wel uniek zijn op grond van de specifieke wijze van configuratie uit standaardcomponenten.

We bespreken in het navolgende twee uitersten met betrekking tot de uniekheid van software, te weten:

- maatwerksoftware
- standaardsoftware.

Twee begrippen staan daarbij centraal: het begrip klantafhankelijkheid en het begrip hergebruik.

### *Maatwerksoftware*

Met maatwerksoftware wordt bedoeld software waarbij men tijdens de ontwikkeling heeft getracht zo volledig mogelijk rekening te houden met de specifieke informatiebehoeften van een bedrijfssituatie. Maatwerksoftware is dus in het uiterste geval uniek. De ontwikkeling van dergelijke software is sterk klantafhankelijk (*de klant specificeert!*). Dit betekent dat gebruikers intensief betrokken zijn bij zowel het definiëren en specificeren van eisen als bij de verificatie en validatie van tussen- en eindproducten. Voor ontwikkelaars ontbreken immers andere soorten referentiemateriaal. Hergebruik is in deze situatie nauwelijks mogelijk. Uitgangspunt is kennis en ervaring in de hoofden van ontwikkelaars en opdrachtgevers of klanten. Dit type softwareontwikkeling staat ook bekend als het verkopen van puur productie- of ontwikkelcapaciteit door ontwikkelaars aan opdrachtgevers (Trienekens en Kusters, 1992). Ontwikkelaars zijn niet gespecialiseerd in bepaalde soorten software. Door ontwikkelaars wordt nauwelijks met ontwikkelstandaards gewerkt. Men tracht zich volledig te richten naar de wensen en de voorkeuren van gebruikers.

In minder extreme vormen van de maatwerkontwikkeling worden referentiemodellen gebruikt. Referentiemodellen weerspiegelen eerder gedaan werk op het gebied van het structureren van de bedrijfsvoering en de informatievoorziening (Greveling, 1990).

### *Kwaliteit van maatwerksoftware*

Het criterium voor kwaliteit is tevredenheid bij de klant. Functionele en technische onvolkomenheden kunnen acceptabel zijn voor ontwikkelaars mits de tevredenheid van de opdrachtgever daar niet onder lijdt. naadloze inpassing van software binnen bedrijfsprocessen zal bijvoorbeeld belangrijker zijn dan hoge eisen aan modulariteit en uitbreidbaarheid van de software.

Ontwikkelaars richten zich sterk op het aftasten van onuitgesproken wensen én behoeften van klanten. Zowel objectieve als subjectieve eisen en wensen tracht men mee te nemen als uitgangspunt bij het ontwerp. Door allerlei, eventueel niet gespecificeerde, extra's kan worden getracht de kwaliteit te verhogen. Dit kan variëren van het zo volledig mogelijk aansluiten van de software op handmatige procedures binnen een bedrijfssituatie tot het vergroten van de aantrekkelijkheid van de software door het aanbrengen van 'toeters en bellen', bijvoorbeeld door extra aandacht te besteden aan kleurgebruik, geluid en vormaspecten.

### *Standaardsoftware*

Standaardsoftware kan in meerdere soortgelijke bedrijfssituaties, bijvoorbeeld bedrijfsbranches, worden toegepast. Ontwikkelaars van standaardsoftware beperken zich over het algemeen tot bepaalde 'productfamilies'. Ontwikkelaars zijn gericht op het ontwikkelen van softwarecomponenten met het oog op hergebruik. De componenten worden in plaats van klantafhankelijk, marktafhankelijk ontwikkeld. Het ontwikkelwerk wordt zoveel mogelijk gestandaardiseerd bijvoorbeeld qua structurering en codering. Van herbruikbaar materiaal bestaan inmiddels vele definities. Bijvoorbeeld door Trienekens en Kusters (1992) wordt onderscheid gemaakt in hergebruik van informele componenten, zoals bijvoorbeeld referentiemodellen, en in hergebruik van formele componenten, zoals bijvoorbeeld programmaprocedures, database-structuren, en mensmachinedialogen. Ook wordt onderscheid gemaakt in herbruikbare product- en procescomponenten. Voorbeelden van procescomponenten zijn: levenscyclusmodellen (bijvoorbeeld: lineair, incrementeel, prototyping) maar ook gedetailleerde activiteitenstructuren (bijvoorbeeld voor het normaliseren van gegevensstructuren).

Doel van hergebruik is om op efficiënte wijze op basis van een gelimiteerd aantal componenten of bouwstenen varianten binnen voorgedefinieerde productfamilies te ontwikkelen. Meerdere softwareproducten worden parallel aan elkaar ontwikkeld, zoveel als mogelijk gebaseerd op dezelfde basiscomponenten.

Er bestaan verschillende vormen van standaardsoftware. Bijvoorbeeld een softwarepakket dat parameteriseerbaar is binnen een bepaalde bedrijfssituatie voor een bepaald soort productiebesturing. Een ander voorbeeld is een standaardpakket dat als infrastructurele component organisatiebreed wordt verspreid, bijvoorbeeld een tekstverwerkingspakket, een spreadsheetapplicatie etc.).

#### *Kwaliteit van standaardsoftware*

Formeel hergebruik van componenten bij de ontwikkeling van standaardsoftware is slechts mogelijk wanneer bouwstenen met het oog op hergebruik worden ontwikkeld. Dit betekent dat deze bouwstenen een hoge graad van perfectie dienen te hebben. Zwakke plekken in componenten kunnen immers in een (nieuwe) configuraties fataal zijn. Standaardisatie en modularisatie van bouwstenen zijn belangrijk kwaliteitseisen voor deel- of tussenproducten van het ontwikkelwerk. Deze kenmerken vormen een voorwaarde voor het voldoen van eisen zoals flexibiliteit, koppelbaarheid en overdraagbaarheid.

Het voorgaande toont aan dat softwarekwaliteit varieert met de mate van uniekheid van de software. opdrachtgevers dienen zich te realiseren dat men in geval van maatwerksoftware in feite in staat moet zijn de gewenste en vereiste kwaliteitskenmerken zelf te identificeren en te specificeren. in geval van standaardsoftware is men, in elk geval gedeeltelijk, afhankelijk van de visie, de doelstellingen en de vaardigheden van de ontwikkelaar.

In deze paragraaf is beschreven dat software op verschillende manieren kan worden beschouwd. De drie invalshoeken die we hebben gehanteerd zijn respectievelijk:

- de productbegrenzing van software
- de toestanden die software, tijdens de levenscyclus doorloopt
- de uniekheid van software.

Met voorbeelden is toegelicht dat binnen elke invalshoek de definitie van softwarekwaliteit verschuift en dat daarmee verschillende kwaliteitskenmerken naar voren komen.

## **6.5 Kwaliteitskenmerken zijn afhankelijk van actoren**

Ook in het boek van Van Zeist e.a. (1996) wordt nader ingegaan op het thema ‘verschillende situaties, verschillende kwaliteitskenmerken’. Van Zeist e.a. gaan echter uit van de gezichtspunten van verschillende typen personen die zijn betrokken bij software(-ontwikkeling) en trachten vanuit de karakteristieken van die personen aan te geven welke kwaliteitskenmerken meer of minder relevant zijn.

De typen personen die worden onderscheiden zijn respectievelijk:

- *de gebruiker*, onderverdeeld in respectievelijk de eindgebruiker, de koper (gebruikersmanager) en de gebruikersondersteuner
- *de ontwikkelaar*, onderverdeeld in productontwerper, productontwikkelaar, productbeheerder en projectmanager
- *de evaluator*, onderverdeeld in de productevaluator, de certificeerder en de EDP-auditor.

We lichten dit toe aan de hand van enkele voorbeelden

NB: voor de Engelstalige definities van kwaliteitskenmerken zie de betreffende Appendix 2. De begrippen die in deze appendix worden gepresenteerd, worden in de oorspronkelijke Engelstalige omschrijving gegeven. Dit is bewust gedaan omdat de begrippen na veel wiken en wegen door de ISO/IEC uiterst zorgvuldig zijn gedefinieerd. Bij een vertaling naar het Nederlands zou een deel van deze zorgvuldigheid verloren kunnen gaan..

#### *Eindgebruiker*

Vanuit het gezichtspunt van de eindgebruiker moet het softwareproduct zijn dagelijkse werk ondersteunen. Het moet de gebruiker helpen bij het uitvoeren van zijn taken en moet die vereenvoudigen. Kleine ongemakken kunnen resulteren in grote ergernis. De consequentie van dit gezichtspunt voor de inhoud van een specificatie van kwaliteitskenmerken is dat over het algemeen een sterke nadruk ligt op usability-kenmerken, bijvoorbeeld het subkenmerk attractiveness. Daarnaast zijn efficiency kenmerken, met name time-behaviour, van belang. Binnen het kwaliteitskenmerk functionality zal vaker het subkenmerk suitability hoog op de ranglijst staan.

#### *Koper (gebruikersmanager)*

Een koper zal willen beschikken over een duidelijke en volledige kwaliteitsspecificatie. Een belangrijk doel van een koper, in een rol van gebruikersmanager, is tevredenheid en instemming onder de gebruikers. De aandacht vanuit dit gezichtspunt gaat meestal in de richting van functionality- en efficiency-kenmerken. Door een manager kan bijvoorbeeld een risico-analyse worden gedaan om het belang van bepaalde kenmerken te bepalen. Vaak zijn portability-kenmerken minder van belang, tenzij de manager verantwoordelijk is voor gebruikers die op heterogene systeemplatformen werken.

#### *Gebruikersondersteuner*

Dit gezichtspunt treffen we aan bij personen die gebruikers assisteren met het werken met software. Wijzigen, aanpassen aan wensen en installeren, maken onderdeel uit van de primaire taken onderhoud en ondersteuning. Binnen gebruikersondersteuning wordt vaak veel nadruk gelegd op portability-kwaliteitskenmerken zoals installability en adaptability, en maintainability-eigenschappen zoals stability en manageability.

#### *Ontwikkelaar (engineer, ontwerper, projectmanager, beheerder)*

Een ontwikkelaar is verantwoordelijk voor het verwerken van alle software-eisen in het software-ontwerp. Vanuit dit gezichtspunt bestaat grote behoefte aan duidelijke en praktisch hanteerbare maatregelen die kunnen worden genomen om kwaliteitskenmerken in te bouwen in de software. Daarnaast zijn metrieken noodzakelijk om te kunnen vaststellen of bepaalde maatregelen tot de gewenste effecten hebben geleid. Voor de productbeheerder wordt wel expliciet een belangrijk kwaliteitskenmerk genoemd, te weten maintainability, met name analysability en traceability.

#### *Evaluator*

De gezichtspunten van de evaluator zijn gebaseerd op verschillende soorten productevaluaties. Respectievelijk worden onderscheiden:

- De algemene productevaluatie, een product wordt geëvalueerd om het te kunnen vergelijken met andere producten. Praktijkonderzoek wijst uit dat de algemene productevaluator

zich in de praktijk voornamelijk concentreert op functionality, met bijzondere aandacht voor ‘geschiktheid voor gebruik’ ofwel suitability.

Een certificatie, een product wordt gecertificeerd om de producent of de kopers vertrouwen te geven in de kwaliteit. Voor een certificeerde zijn met name kwaliteitskenmerken zoals conformance en compliance van belang.

Een EDP-audit, een evaluatie door een EDP-auditor wordt ook wel systeem-audit genoemd. Een EDP-audit kan worden verricht op verzoek van een accountant als ondersteuning van diens accountantscontrole. Voor een EDP-auditor staat het kwaliteitskenmerk reliability centraal. De controleerbaarheid en de beheersbaarheid van bijvoorbeeld verwerkingsprocessen binnen of door de software zijn een belangrijk aandachtspunt. Ook beschikbaarheid van (informatie die wordt gegenereerd door) de software is een relevant kwaliteitskenmerk.

Bovenstaande laat zien dat vanuit verschillende gezichtspunten verschillende kwaliteitskenmerken op de voorgrond kunnen worden gesteld. In discussie over gewenste en/of noodzakelijke kwaliteit is het daarom raadzaam eerst duidelijk te krijgen met welke betrokken personen men die discussie voert, en wat hun belangen en hun taken zijn. Op grond daarvan kan wellicht op een meer efficiënte en effectieve wijze een specificatie van kwaliteitskenmerken tot stand komen.

## 6.6 Conclusies

In dit hoofdstuk hebben we ons gericht op een nadere beschrijving van het begrip softwarekwaliteit. Nadat we een aantal bekende opvattingen en de verschillen en overeenkomsten tussen deze opvattingen de revue hebben laten passeren, hebben we de meest recente ‘state of the art’ definitie van softwarekwaliteit behandeld, namelijk de definitie volgens ISO/IEC 9126. Deze standaard, die weliswaar nog in ontwikkeling is, laat met via het begrip ‘kwaliteit-in-gebruik’ zien dat kwaliteit *kan* of beter *moet* worden afgeleid uit de bedrijfs- en/of gebruiksomgeving waar software wordt toegepast. Door ISO/IEC wordt een contingentieprincipe naar voren gebracht: dat wil zeggen dat verschillende situaties vragen om verschillende kwaliteitskenmerken.

In paragraaf 6.4 hebben we, dit principe indachtig, uitgelegd dat afhankelijk van de beschouwingswijze van software, in een specifieke situatie verschillende kwaliteitskenmerken een rol spelen. Een en ander maakt duidelijk dat softwarekwaliteit en daarmee kwaliteitskenmerken moeten worden afgeleid uit de bedrijfs- of gebruiksomgeving in kwestie. In hoofdstuk 7 wordt deze opvatting verder uitgewerkt en worden benaderingen beschreven voor het afleiden van kwaliteitskenmerken uit de behoeften van gebruikers in een bepaalde bedrijfsomgevingen.