Softwaremetingen binnen het Capability Maturity Model

Drs. C.M. Piek

Het Capability Maturity Model (CMM) is door de software-industrie als kader algemeen geaccepteerd om de kwaliteit van haar softwareprocessen (en in het verlengde hiervan haar softwareproducten) continu te kunnen verbeteren. Deze verbeteringen dienen echter gepaard te gaan met softwaremetingen. Hiervoor is een methode nodig die binnen het CMM gebruikt kan worden om de geleverde 'kwaliteit' van een softwareproces te meten.

Inleiding

De jaren negentig laten zich kenmerken door de bewustwording in de software-industrie dat continue kwaliteitsverbetering van doorslaggevende betekenis kan zijn om op de langere termijn te kunnen overleven. Dit kan onder andere verklaard worden doordat de 'markt' steeds hogere kwaliteitseisen stelt aan softwareproducten.
Voor een blijvend betrouwbare werking van de informatievoorziening dienen onder andere softwareproducten van hoge kwaliteit te zijn. Daardoor worden softwareorganisaties gedwongen om kwaliteitssystemen te ontwerpen en in te richten die kwalitatief hoogwaardige softwareproducten kunnen voortbrengen. Een kwaliteitssysteem dat ontworpen is voor softwareorganisaties is het Capability Maturity Model van Humphrey ([Hump89]).
Het CMM wordt dan ook door softwareorganisaties gebruikt om verbeteringen in hun softwareprocessen te kunnen aanbrengen. Het merendeel van deze organisaties bevindt zich momenteel op CMM level 1 en er worden diverse acties ondernomen om dit niveau te ontstijgen ([Sass96]). Om kwaliteitsverbeteringen te kunnen aanbrengen, dient de organisatie kwaliteitskringlopen te doorlopen om uiteindelijk op een hoger gelegen CMM-niveau te kunnen komen ([Demi82], [Hump89]).

Tijdens het incrementeel doorlopen van deze kringlopen kunnen softwaremetingen en softwarekwaliteitsmetrieken een organisatie helpen. Om verbeteringen in de ontwikkeling en het onderhoud van softwareproducten te kunnen realiseren, is het voor een organisatie noodzakelijk inzicht te verkrijgen in die factoren die de softwarekosten en -activiteiten beïnvloeden. Software(kwaliteits)metrieken kunnen dit inzicht geven. Door het definiëren en analyseren van metrieken en bijbehorende metingen die betrekking hebben op softwareactiviteiten kan een organisatie inzicht krijgen om verbeteringen door te voeren in zowel de kwaliteit van het softwareproces als het softwareproduct zelf ([Grad87], [Neil90]). Echter, organisaties ondervinden problemen bij het toepassen van (bestaande) softwaremeetmethoden met betrekking tot het definiëren en implementeren van een stelsel van metrieken die specifiek zijn afgestemd op de karakteristieken van de softwareontwikkelings/onderhoudsomgeving en de behoeften van de betreffende organisatie ([Roch94]).

Naast de methode die in dit artikel zal worden gepresenteerd, waarmee binnen het CMM een willekeurig softwareproces kan worden gemeten, zal de methode worden toegepast op het correctieve onderhoudsproces van een softwareleverancier.

Capability Maturity Model

Het CMM maakt onderscheid tussen een vijftal niveaus van volwassenheid waarop een softwareorganisatie zich kan bevinden: Initial, Repeatable, Defined, Managed of Optimizing. Het onderscheid dat in dit model tussen de verschillende niveaus wordt aangebracht, berust op het idee dat het ene softwareproces volwassener is dan een ander softwareproces. Tevens wordt op basis van ervaringskennis uiteengezet hoe een softwareorganisatie van het ene niveau naar een hoger gelegen niveau kan 'groeien'.

CMM-fundamenten

Voor een beter begrip van het CMM dient ten eerste een aantal definities te worden gegeven ten aanzien van softwareprocessen ([Paul93]):
* *Software proces capability* is de mate waarin bepaalde voorgedefinieerde uitvoer kan worden verwacht wanneer het betreffende softwareproces wordt uitgevoerd.
* *Software process performance* is de prestatie waarmee processen bepaalde uitvoer voortbrengen. Derhalve richt de *process performance* zich op de prestaties zoals die door het betreffende proces worden geleverd, terwijl de *process capability* zich richt op de resultaten dan wel de uitvoer die van dit proces mag worden verwacht.
* *Software process maturity* is de mate waarin het softwareproces kan worden gedefinieerd, gestuurd, gemeten en gecontroleerd. Hierbij dient te worden opgemerkt dat naarmate de *volwassenheid* van het softwareproces toeneemt eveneens de *process capability* en de *process performance* zullen toenemen.

CMM: vijf niveaus van volwassenheid

Zoals gesteld kent het CMM vijf niveaus van volwassenheid.

Niveau 1: Initial software process
Het eerste niveau, het *initial* proces, wordt gekarakteriseerd door een ad-hocbenaderingswijze van het softwareproces. De eisen waaraan de invoer van het proces moet voldoen, is niet of slecht gedefinieerd. Tevens wordt er een bepaalde uitvoer verwacht en heeft de organisatie geen inzicht in de transformatie van de invoer naar de uitvoer. Hierom worden de softwareprocessen op dit niveau *initial* genoemd.

Niveau 2: Repeatable software process
Door de in het verleden opgedane ervaringen in het ontwikkelen en onderhouden van min of meer dezelfde softwareproducten heeft de organisatie een zekere statistische controle over kosten en tijdschema's met betrekking tot het ontwikkelen van softwareproducten verkregen. Hierom worden de softwareprocessen op dit niveau *repeatable* genoemd.

De introductie van nieuwe softwaretools, de ontwikkeling van een nieuw type softwareproduct en belangrijke organisatorische veranderingen behoren tot de belangrijkste risicogebieden op dit niveau. De hoofdactiviteiten om dit niveau te ontstijgen (naar CMM-niveau 3) zijn:
* oprichten van een softwareprocesgroep. Deze groep is een kleine eenheid die het verbeteren van het softwareproces als enige (primaire) taak heeft.
* beschrijven van een softwareprocesmodel. Met behulp van het in de paragraaf 'Softwareprocesmodel' beschreven procesmodel kan een raamwerk, waarbinnen specifieke ontwikkelings- en onderhoudsprocessen zijn gedefinieerd, in meetbare termen worden gemodelleerd.
* de introductie van een verzameling softwaremethoden en -technieken, zoals te gebruiken standaarden, ontwerp- en testmethoden en programmacode-inspectietechnieken.

Niveau 3: Defined software process
Op CMM-niveau 3 zijn de activiteiten gedefinieerd waaruit het softwareproces is opgebouwd. Tevens zijn de eisen gedefinieerd waaraan de invoer en de uitvoer van deze deelprocessen moeten voldoen. Hierom worden de softwareprocessen op dit niveau *defined* genoemd.
Op dit niveau heeft de organisatie een basis bereikt om continu verbeteringen door te kunnen voeren. Door het beoordelen en analyseren van de bestaande softwareprocessen kunnen probleemgebieden worden geïdentificeerd om verbeteringen te kunnen aanbrengen. De hoofdactiviteiten die op dit niveau onderscheiden kunnen worden, zijn:
* definiëren van een verzameling softwaremetrieken en het ontwerpen van een database om de gegevens die tijdens het softwareproces worden vastgelegd, te kunnen verzamelen en te onderhouden;
* opstellen van een softwarekwaliteitsplan waarin kwaliteitsdoelen zijn geformuleerd. De mate waarin deze doelstellingen zijn gerealiseerd, wordt beoordeeld door kwantitatieve metingen uit te voeren.

Niveau 4: Managed software process
Organisaties hebben op dit niveau kwantificeerbare kwaliteitsdoelstellingen geformuleerd voor zowel hun softwareproducten als -processen. In een softwaremeetprogramma wordt beschreven hoe de doelstellingen dienen te worden gemeten. In een organisatiebrede database worden de gegevens vastgelegd om de bewuste metingen te kunnen uitvoeren. Deze metingen vormen een kwantitatieve basis om softwareprocessen en -producten te kunnen evalueren. Tevens wordt een organisatie in staat gesteld trends in de kwaliteit van haar softwareproducten en -processen te kunnen voorspellen binnen voor de organisatie acceptabele grenzen.

Niveau 5: Optimizing software process
Op dit niveau heeft de softwareorganisatie continu aandacht voor softwareprocesverbeteringen. Softwaremetrieken betreffende de kwaliteit van het softwareproces worden gebruikt voor kosten-batenanalyses op nieuw toegepaste technologie. Technologische (software-)innovaties en methoden die resulteren in kwaliteitsverbeteringen worden als zodanig geïdentificeerd en gebruikt om veranderingen in het proces aan te brengen.
Defecteninformatie wordt gebruikt om defecten te analyseren en de oorzaken hiervan te onderzoeken. De bevindingen uit de oorzakenanalyse, in de vorm van leerprocessen, worden teruggekoppeld naar het betreffende softwareproces. Het doel hiervan is om het optreden van soortgelijke defecten in de toekomst tot een minimum te reduceren.

Samenvattend kan worden gesteld dat naarmate een softwareorganisatie volwassener wordt (een hoger gelegen CMM-niveau bereikt), softwareproducten tegen lagere kosten, met minder risico's en tegen hogere kwaliteit kunnen worden ontwikkeld.

Softwarekwaliteitskringloop

Een softwareproduct doorloopt verschillende fasen tijdens zijn levenscyclus. Afhankelijk van de fase waarin een softwareproduct verkeert en waarvoor men verantwoordelijk is, varieert de betekenis van productkwaliteit ([Trie94]). Nadat het softwareproduct op de markt is gebracht, komt het in de onderhoudsfase van zijn levenscyclus terecht. Tijdens de onderhoudsfase ondergaat het softwareproduct veranderingen, waarbij de kwaliteitseigenschap *onderhoudbaarheid* een grote betekenis heeft voor degene die onderhoud pleegt op het softwareproduct in kwestie.
De kwaliteitskringloop is een theoretisch model van de onderlinge wisselwerking tussen de kwaliteit van het softwareproces en de kwaliteit van een softwareproduct die in de verschillende fasen kan worden onderkend. In het bijzonder zal in deze paragraaf de onderhoudsfase nader worden belicht.

Kwaliteit, verschillende invalshoeken

'De kwaliteit van een object (bijvoorbeeld van een softwareproduct of IT-dienst) is het geheel van kenmerken van dat object die van invloed zijn op zijn vermogen, te voldoen aan vastgelegde en vanzelfsprekende behoeften' ([ISO94]). De invalshoek die wordt ingenomen bij het definiëren van het begrip kwaliteit bepaalt de manier

waarop kwaliteit door de belanghebbende wordt beoordeeld. Garvin ([Garv84]) concludeert dat kwaliteit een complex en multidimensionaal concept is, dat kan worden beschreven vanuit vijf verschillende invalshoeken: (1) de transcendente, (2) de gebruikersgeoriënteerde, (3) de productiegeoriënteerde, (4) de productgerichte en (5) de waardegerichte invalshoek. Hieronder zullen alleen de gebruikers-, de productiegeoriënteerde en de productgerichte invalshoek worden beschreven.

Gebruikersgeoriënteerde invalshoek

Juran ([Jura70]) definieert de gebruikersgeoriënteerde invalshoek van kwaliteit als 'fitness for use'. De klant verwacht dat het softwareproduct geschikt is voor gebruik. Deze geschiktheid wordt bepaald door de mate waarin het softwareproduct zijn primaire en/of secundaire bedrijfsprocessen ondersteunt. Ofwel de mate waarin de kwaliteitseisen, die vanuit de bedrijfsprocessen aan het softwareproduct worden gesteld, terug te vinden zijn in de productkarakteristieken van het product.
Nadat het softwareproduct bij de klant in gebruik is genomen, is het onderhevig aan allerlei krachten als gevolg van onder andere de continu veranderende bedrijfsprocessen. Het gemak en de kosten waarmee het softwareproduct aan de hierdoor veranderende kwaliteitseisen en bedrijfsprocessen aangepast kan worden, bepalen de blijvende 'geschiktheid voor gebruik' van de klant. Ofwel, de klant verwacht:

* *blijvende* betrouwbare werking van het softwareproduct;
* *blijvende* effectieve ondersteuning van het softwareproduct.

Het kwaliteitssysteem van de softwareorganisatie is een afbeelding van het kwaliteitsbeleid.

Productiegeoriënteerde invalshoek

Crosby ([Cros79]) definieert de productiegeoriënteerde definitie van kwaliteit als 'conformance to requirements'. Om een softwareproduct geschikt te maken voor gebruik moet aan de verwachtingen en behoeften van de klant worden voldaan. Deze kunnen in service level agreements (SLA's) neergelegd zijn. De productiegeoriënteerde benadering stelt dat de kwaliteit van het ontwikkelingsproces wordt bepaald door de mate waarin aan deze verwachtingen wordt voldaan. Ofwel, de mate waarin een kwalitatief hoogwaardig eindproduct wordt opgeleverd in termen van bijvoorbeeld het aantal *latente defecten* en de hieraan verbonden kosten gedurende de onderhoudsfase. De productiegeoriënteerde benadering is terug te vinden bij het CMM; het (continu) verbeteren van het softwareproces heeft tot gevolg dat de kwaliteit van het software-eindproduct (tevens continu) verbetert ([Hump89]), ([Paul93]).

Productgerichte invalshoek

In tegenstelling tot de gebruikers- en de productiegeoriënteerde invalshoek die de kwaliteit van buitenaf beschouwen, wordt de productgerichte definitie van kwaliteit van binnenuit beschouwd. De productgerichte definitie van kwaliteit heeft betrekking op de intrinsieke productkarakteristieken die (in principe) kunnen worden gemeten ([Boeh84], [Gilb88]).
Vanuit de invalshoek die ingenomen wordt bij het definiëren van productkwaliteit kunnen verschillende verzamelingen kwaliteitseigenschappen door de belanghebbenden als relevant worden geacht. In de literatuur hebben diverse auteurs overzichten gegeven van wat zij wezenlijke eigenschappen van een softwareproduct vinden. Hierbij hadden de auteurs de intentie om een volledig en verifieerbaar beeld van de kwaliteit van een softwareproduct, voor alle belanghebbenden, te geven.
De kwaliteitsmodellen worden gerepresenteerd door een hiërarchisch stelsel van kwaliteitsattributen, waarbij elk lager niveau een verdere concretisering van het hogere niveau voorstelt. Deze modellen kunnen bijvoorbeeld worden gevonden bij [McCa77], [Boeh78], [Dele90] en [ISO92].

Kwaliteit in beweging

Het softwareproduct is op te vatten als het belangrijkste product van de softwareorganisatie. De productiegeoriënteerde benadering van kwaliteit stelt dat de kwaliteit van het ontwikkelingsproces bepaalt in welke mate een kwalitatief hoogwaardig eindproduct wordt opgeleverd ([Hump89], [Paul93]). Het kwaliteitssysteem van de softwareorganisatie, waarbinnen de kwaliteit van het softwareontwikkelingsproces wordt bepaald, is een afbeelding van het kwaliteitsbeleid.
Bij het formuleren van het kwaliteitsbeleid streeft de softwareorganisatie naar een maximalisatie van bedoelde voorziene effecten: het op de markt brengen van een kwalitatief hoogwaardig softwareproduct. Nadat het softwareproduct op de markt is gebracht en bij de klanten in gebruik is genomen, komt de kwaliteit van het softwareproduct in beweging:

* Als gevolg van storingen, veroorzaakt door defecten in het softwareproduct, ondervindt de klant problemen tijdens het gebruik van het softwareproduct.
* Als gevolg van continue veranderingen in de bedrijfsprocessen neemt de mate waarin het softwareproduct de bedrijfsprocessen ondersteunt geleidelijk aan af.

De softwareorganisatie reageert op deze dalende kwaliteit door middel van het uitvoeren van diverse vormen van onderhoud: correctief, adaptief of perfectief onderhoud. De kwaliteit van de onderhoudsprocessen bepaalt de mate waarin de 'oorspronkelijke productkwaliteit' wordt gehandhaafd.

Verminderende kwaliteit

Er kunnen zich in onder meer de gebruikersorganisatie allerlei situaties voordoen die wijzigingen in het softwareproduct noodzakelijk maken. De softwareorganisatie zal door het uitvoeren van onderhoud een blijvende betrouwbare werking en blijvende effectieve ondersteuning van het softwareproduct garanderen. Door het herhaald uitvoeren van dit onderhoud neemt als gevolg van de 'Laws of Software Evolution' de kwaliteit van zowel het softwareproduct als het softwareproces geleidelijk aan af.

Laws of Software Evolution

Lehman ([Lehm80]) heeft een groot aantal (complexe) softwareproducten onderzocht en heeft vervolgens een vijftal 'Laws of Software Evolution' geformuleerd. Twee van die wetten hebben gevolgen voor de kwaliteit van een softwareproces dan wel -product:

* *Wet van de voortdurende verandering*. De organisatie waarin het softwareproduct operationeel is, is continu onderhevig aan zowel interne als externe veranderingen. Om een blijvende effectiviteit van het softwareproduct te kunnen garanderen zullen tevens veranderingen in het softwareproduct dienen te worden aangebracht.
* *Wet van de toenemende complexiteit*. Door het continu veranderen van het softwareproduct zal de oorspronkelijke structuur in complexiteit toenemen. Denk hierbij aan de legacy systemen die nog steeds bij de grotere ondernemingen operationeel zijn.

Op grond van deze twee wetten neemt zowel de kwaliteit van het softwareproduct als de kwaliteit van het softwareproces af. Als de softwareorganisatie geen pogingen onderneemt om continu verbeteringen in zowel haar softwareproducten als -processen aan te brengen, zal de tevredenheid van de klant in de tijd bezien ook gaan afnemen.

Wet van de softwareaccumulatie

Diverse studies hebben uitgewezen dat complexe softwareproducten meer (correctief) onderhoud vereisen gedurende de onderhoudsfase ([Grem84], [Boeh81], [Lien81]). Complexe softwareproducten bevatten hierdoor meer *latente defecten*, die tijdens het ontwikkelingsproces onopgemerkt zijn gebleven. Figuur 1 laat zien dat aangebrachte wijzigingen dan wel opgeloste problemen nieuwe problemen kunnen introduceren en wijzigingen noodzakelijk kunnen maken ([Looij95], [Valla88]). Ofwel, door het uitvoeren van onderhoud kunnen nieuwe defecten worden geïntroduceerd. Onderhoud is dus moeilijk vanwege de toenemende complexiteit en als gevolg hiervan is meer onderhoud vereist.

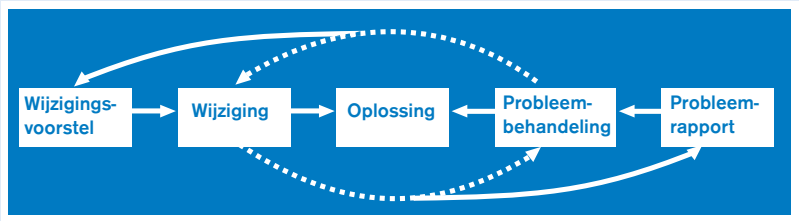
De reactie van de onderhoudsorganisatie op de dalende kwaliteit van het softwareproduct is op te vatten als een proces van *softwareaccumulatie*. Preventief onderhoud moet de neergaande spiraal van dalende kwaliteit doorbreken. Het doel van preventief onderhoud is tweeledig:

* het voorkomen van toekomstige problemen dan wel defecten;
* het verhogen van de kwaliteit van het softwareproduct.

(Continue) kwaliteitsverbetering

Om kwaliteitsverbeteringen te kunnen aanbrengen is het nodig het geleverde kwaliteitsniveau te registreren (in de vorm van onder meer probleem- en wijzigingsrapporten), defecten te analyseren en de oorzaken hiervan te onderzoeken. De bevindingen uit de oorzakenanalyse, onder andere in de vorm van leerprocessen, worden teruggekoppeld naar het betreffende kwaliteitssysteem of zelfs naar het kwaliteitsbeleid van de softwareorganisatie (zie figuur 2).

Kwaliteitsverbetering begint met het verkrijgen van inzicht in de wisselwerking tussen de kwaliteit van het



Figuur 1. Wijzigings- en probleemspiraal ([Looij95]).

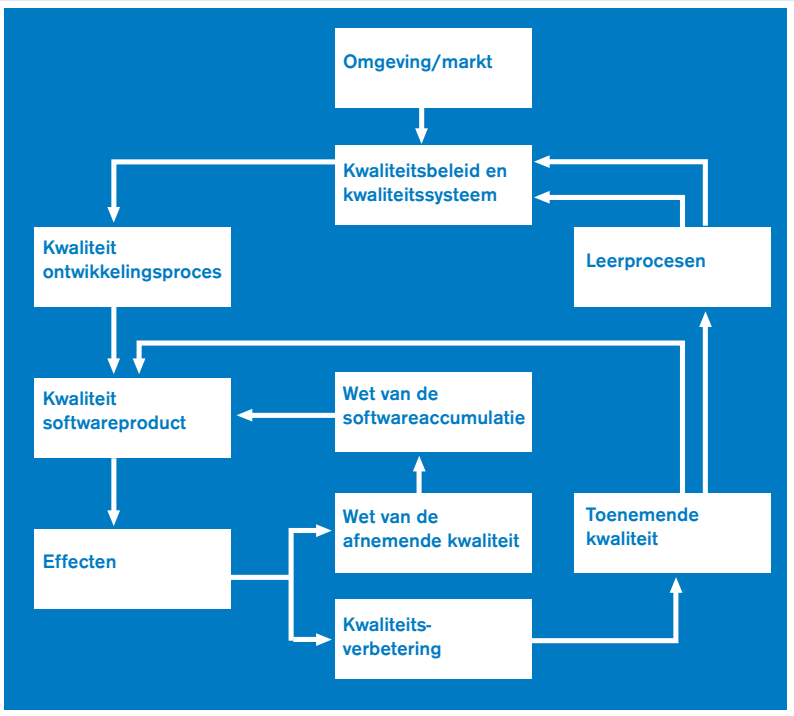
softwareproduct en de kwaliteit van het softwareproces ([Grad87], [Neil90]). Deze *enkelvoudige* kwaliteitskringloop is in deze paragraaf beschreven. Om de betreffende fase te kunnen bewaken en te beoordelen (dat wil zeggen het uitvoeren van kwaliteitscontroles op de software-entiteiten die in een softwareomgeving zichtbaar aanwezig zijn) en waar nodig verbeteringen door te kunnen voeren is een model nodig dat de bewuste fase in meetbare termen beschrijft ([Fent91]). Dit model zal in de volgende paragraaf worden beschreven.

Kwantitatief modelleren van de softwarekwaliteitskringloop

Het softwareproduct doorloopt verschillende fasen tijdens zijn levenscyclus. Afhankelijk van de fase waarin het softwareproduct zich bevindt zal kwaliteit door de belanghebbenden uit verschillende aspecten zijn opgebouwd. Om deze aspecten te kunnen kwantificeren dient de softwareomgeving van de organisatie in meetbare termen te worden gemodelleerd ([Fent91]).

Softwareprocesmodel

Een organisatie dient haar toegevoegde waarde te maximaliseren door zowel kwalitatief hoogwaardige softwareproducten te leveren als haar softwareprocessen optimaal af te stemmen op haar omgeving. Om de klan-



Figuur 2. Softwarekwaliteitskringloop.

ten van een softwareproces optimaal te kunnen bedienen moet de organisatie de geleverde 'kwaliteit' van de in de softwareomgeving aanwezige software-entiteiten continu bewaken en beoordelen. Deze benadering is terug te vinden in het CMM; het (continu) verbeteren van de kwaliteit van het proces heeft tot gevolg dat de kwaliteit van het softwareproduct ook (continu) verbetert.

Fundamenten softwareproces

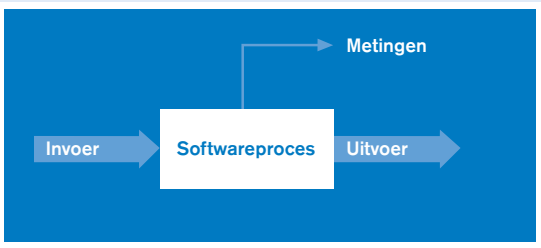
Alvorens het softwareprocesmodel zelf te behandelen dient een aantal reeds algemeen geaccepteerde aannames te worden gemaakt met betrekking tot dit proces ([Hump89]):

* Het softwareproces, dat opgedeeld kan worden in deelprocessen, kan worden gecontroleerd, gemeten en verbeterd.
* Door het gestructureerd uitvoeren van het softwareproces kunnen geplande kostenramingen, tijdsplanningen en kwaliteitsniveaus worden gehaald. Dit betekent dat de uitvoer van een softwareproces, binnen redelijke grenzen, kan worden voorspeld. Wanneer het softwareproces statistisch onder controle is, zullen herhaald uitgevoerde soortgelijke softwareprocessen dezelfde uitvoer opleveren.
* Wanneer de organisatie het softwareproces statistisch onder controle heeft, kunnen kwalitatief betere softwareproducten slechts worden verkregen door verbeteringen in het softwareproces aan te brengen. Als het softwareproces niet statistisch kan worden gecontroleerd, kunnen ook geen verbeteringen worden aangebracht.

Generiek softwareprocesmodel

Aangezien softwareprocesmodellen kunnen worden beschreven op een willekeurig abstractieniveau, dient gebruik te worden gemaakt van specifieke modelleermethoden en -technieken. Deze methoden en technieken dienen elementen, standaarden en structurele raamwerken aan te reiken waarmee door stapsgewijze verfijning gedetailleerde softwareprocesmodellen kunnen worden beschreven. De mate van gedetailleerdheid waarmee de softwareprocesmodellen kunnen worden beschreven, is afhankelijk van de *volwassenheid* van de organisatie die softwareproducten ontwikkelt en onderhoudt.

Figuur 3 toont een generiek softwareprocesmodel ([Hump89]). Het model laat zien dat het uitvoeren van een softwareproces, gegeven een bepaalde invoer en gebruikmakend van de hiervoor benodigde middelen, een bepaalde uitvoer oplevert. Het model uit figuur 3 kan zowel het gehele softwareproces voorstellen als slechts een gedeelte hiervan, bijvoorbeeld het testen van het softwareproduct. Door het aaneenschakelen van een aantal softwareprocesmodellen kan een logisch samenhangend softwareproces op willekeurig abstractieniveau worden beschreven.



Figuur 3. Generiek softwareprocesmodel.

Fenton ([Fent91]) stelt dat onderstaande drie categorieën software-entiteiten in een generiek softwareprocesmodel kunnen worden gemeten:

* *softwareprocessen*. Tot een softwareproces behoren alle softwareactiviteiten die door de tijd heen plaatsvinden en die uitvoer creëren die van waarde is voor de klant (van het proces). Hierbij is een activiteit een verzameling van gebeurtenissen die totstandkomt onder verantwoordelijkheid van één actor.
* *softwareproducten*. Tot de softwareproducten behoren alle tastbare voorwerpen die voortgebracht worden door een softwareproces. Hiertoe worden dus tevens niet-softwareproducten gerekend. Voorbeelden van niet-softwareproducten zijn probleem- en wijzigingsrapporten, meetgegevens en handleidingen.
* *middelen*. Tot de middelen behoren alle zaken die nodig zijn om een proces uit te kunnen voeren, met uitzondering van de (niet-)softwareproducten voortgebracht door een ander proces. Voorbeelden van middelen zijn personeel, softwaretools en computers.

Softwaremetrieken en softwareprocesmodel

Het gebruik van softwaremetrieken helpt een organisatie bij het eenduidig definiëren van de manier waarop een attribuut kan worden gemeten. Wel dient dan bekend te zijn van welke entiteit de metriek een indicatie geeft over de mate van bezit van een eigenschap. Een metriek dient gerelateerd te zijn aan een meetbare software-entiteit. Een en ander maakt het noodzakelijk om de softwarekwaliteitsmetrieken te integreren met het softwareproces ([Bhid90], [Busc90], [Fent91], [Henr96]). In navolging van het door Fenton aangebrachte onderscheid in meetbare software-entiteiten kunnen softwaremetrieken in de volgende drie categorieën worden ingedeeld:

* *productmetrieken,* die attributen van het softwareproduct kwantificeren, bijvoorbeeld kosten, kwaliteit of grootte van een product;
* *procesmetrieken,* die attributen van het softwareproces kwantificeren, bijvoorbeeld kosten, doorlooptijd of aantallen voorkomens van een proces;
* *middelmetrieken,* die attributen van de middelen die nodig zijn om het proces te kunnen uitvoeren, kwantificeren, bijvoorbeeld kosten, kwaliteit of grootte van een middel.

Door de metrieken te relateren aan het softwareprocesmodel kunnen de volgende typen metrieken worden onderscheiden ([Bhid90]):

* *pre-procesmetrieken,* die attributen kwantificeren van software-entiteiten die als invoer van het softwareproces dienen;
* *in-procesmetrieken,* die attributen kwantificeren van software-entiteiten die tijdens de uitvoering van het softwareproces veranderingen ondergaan;
* *post-procesmetrieken,* die attributen van software-entiteiten kwantificeren betrekking hebbende op de uitvoer van het softwareproces.

De softwareorganisatie dient kwaliteit tijdens de gehele levenscyclus van een softwareproduct in het oog te houden. Voor de verschillende fasen definieert de softwareorganisatie hiertoe softwarekwaliteitsmetrieken die betrekking hebben op het geleverde kwaliteitsniveau van

het softwareproces. Softwarekwaliteitsmetrieken die onderdeel uitmaken van softwaremetrieken kwantificeren kwaliteitsattributen van software-entiteiten.
Softwarekwaliteitsmetrieken kunnen, analoog aan bovengenoemde indeling, in drie categorieën worden onderverdeeld en in het model worden ondergebracht ([Kan95]). Door de relaties te onderzoeken tussen de pre-proces- en in-proceskwaliteitsmetrieken en de post-proceskwaliteitsmetrieken kunnen (kwaliteits)verbeteringen worden aangebracht in zowel de softwareprocessen als de softwareproducten. Hierbij dient rekening te worden gehouden met de karakteristieken van de softwareomgeving, en wel met name met het CMM-niveau waarop de organisatie zich bevindt.

Softwaremetrieken en volwassenheid van het proces

In de literatuur zijn vele software(kwaliteits)metrieken beschreven, zie bijvoorbeeld [Fent91], [Grad92] en [Kan95]. De hieruit gekozen verzameling metrieken bepaalt de mate waarin attributen van de binnen het softwareprocesmodel geïdentificeerde entiteiten kunnen worden gekwantificeerd.
Echter, attributen van de te meten entiteiten dienen wel 'zichtbaar' te zijn. De mate waarin attributen 'zichtbaar' zijn, wordt bepaald door de mate van *volwassenheid* van het softwareproces ([Pfle90]). Het CMM biedt hiervoor een goede context ([Hump89], [Paul93]).

Studies uitgevoerd door het Software Engineering Institute wijzen uit dat het merendeel van de softwareorganisaties zich op één van de eerste drie CMM-niveaus bevindt ([Sass96]). Hierom zullen alleen deze niveaus hieronder worden besproken.

CMM en softwaremetrieken

Het principe van de mate van volwassenheid van een softwareproces vindt zijn oorsprong in het CMM ([Hump89], [Paul93]). Naarmate een proces *volwassener* wordt, neemt de mate van gedetailleerdheid waarmee de organisatie haar softwareprocesmodel kan beschrijven steeds verder toe.
Softwaremetingen hangen nauw samen met *zichtbaarheid*; een organisatie kan slechts dat meten wat zichtbaar voor haar is. Het definiëren en analyseren van softwaremetrieken helpt een organisatie bij het begrijpen van die factoren die van invloed zijn op de *performance* en *capability* van het bewuste softwareproces. Het hierdoor verkregen inzicht maakt het voor een organisatie mogelijk het softwareproces steeds beter te kunnen definiëren, beheersen, controleren en (waar nodig) te sturen. Naarmate het proces *volwassener* wordt kunnen additionele metrieken worden gedefinieerd en geanalyseerd.
Derhalve dient de organisatie met het definiëren van metrieken te beginnen op niveau 1 om vervolgens stapsgewijs, door het incrementeel doorlopen van de *enkelvoudige* kwaliteitskringloop, additionele metrieken te definiëren zodat de organisatie uiteindelijk op niveau 5 uitkomt.

Initial software process

De uitgevoerde processen op dit niveau laten zich karakteriseren door een grote verscheidenheid aan bijvoorbeeld productiviteit, planningen en opgeleverde kwaliteit van het (intermediair) eindproduct. Een en ander is het gevolg van het ontbreken van structuur in en een matige definiëring van het softwareproces alsmede het ontbreken van controle- en stuurmaatregelen. Hierdoor zal het uitvoeren van softwaremetingen slechts ten dele (of helemaal niet) kunnen plaatsvinden ([Pfle90]). Organisaties die zich op dit niveau bevinden, dienen de nadruk te leggen op het aanbrengen van structuur in hun softwareprocessen en het definiëren van eisen waaraan de invoer en de uitvoer van een proces moeten voldoen.

Repeatable software process

Op dit niveau zijn de invoer, de uitvoer en de middelen geïdentificeerd. De middelen die op dit niveau kunnen worden onderscheiden, zijn bijvoorbeeld personeel, softwaretools en computers. De organisatie heeft echter geen inzicht in de manier waarop de invoer wordt getransformeerd tot uitvoer. Dat wil zeggen, in de manier waarop een softwareproduct wordt ontwikkeld dan wel onderhouden. Figuur 4 toont het model van het *repeatable* softwareproces.

Figuur 4. Repeatable softwareproces.

Figuur 4 laat tevens zien dat op dit niveau alleen metrieken kunnen worden gedefinieerd betrekking hebbende op de invoer, de uitvoer en de middelen, met uitzondering van softwareproductmetrieken ([Pfle90]). Fenton noemt de invalshoek die op CMM-niveau 2 wordt ingenomen om attributen van het proces te kunnen meten, de *externe invalshoek* ([Fent91]). Deze procesgeoriënteerde invalshoek richt zich op het meten van attributen die betrekking hebben op het softwareproces.

Defined software process

In tegenstelling tot niveau 2 bestaat op niveau 3 inzicht in het softwareproces. Op dit niveau kunnen de activiteiten waaruit het proces is opgebouwd, onderscheiden worden. Tevens zijn de eisen waaraan de onderlinge invoer en uitvoer van deze activiteiten moeten voldoen, bekend en gedefinieerd. Dit betekent dat de onderlinge invoer en uitvoer, die zijn op te vatten als intermediaire softwareproducten, beoordeeld en gemeten kunnen worden. Aan de op CMM-niveau 2 gedefinieerde verzameling metrieken kunnen, op dit niveau, productmetrieken worden toegevoegd ([Pfle90]).
Fenton noemt de invalshoek die op CMM-niveau 3 wordt ingenomen om attributen van het softwareproces te kunnen meten, de *interne invalshoek* ([Fent91]). Deze productgerichte invalshoek richt zich tevens op het meten van productattributen die van invloed zijn op de verschillende activiteiten (waaruit het proces is opgebouwd). Door het meten van productattributen kunnen voorspellingen worden gedaan over hieraan gerelateerde procesattributen. Bijvoorbeeld, door het meten van de complexiteit van een softwareproduct kunnen voorspellingen worden gedaan over de inspanning die nodig is om een defect te kunnen lokaliseren ([Evan95]).

Softwaremetingen

Organisaties die, beginnende op CMM-niveau 1, een aantal malen de *enkelvoudige* softwarekwaliteitskringloop hebben doorlopen, komen uiteindelijk op CMM-niveau 2 terecht. Het bij CMM-niveau 2 behorende kwaliteitssysteem biedt voldoende basis om te beginnen met het definiëren van softwarekwaliteitsmetrieken die betrekking hebben op de *externe invalshoek* van het softwareproces.

Wanneer de organisatie door het uitvoeren van softwaremetingen iets wil weten over een bepaalde fase, dient te worden begonnen met het verzamelen van ruwe gegevens. Dit zijn gegevens die direct van een softwareentiteit zijn af te leiden. Het verzamelen van deze ruwe gegevens is op te vatten als het laagste niveau van het direct meten van entiteiten. De directe softwaremetingen bestaan dan uit het maken van selecties op deze ruwe gegevens om verfijnde gegevens te kunnen verkrijgen.

Figuur 5 laat de rol van het verzamelen van *ruwe* gegevens zien in relatie tot het uitvoeren van softwaremetingen ([Mell92]). Het resultaat van een softwaremeting, verkregen door het toepassen van een geschikte softwaremetriek, is een aan een attribuut toegekende waarde. De verkregen waarde is afgeleid van de verfijnde gegevens, die op hun beurt zijn afgeleid van de verzamelde ruwe gegevens. Derhalve zijn alle softwaremetingen die een organisatie wil uitvoeren, afhankelijk van de ruwe gegevens die tijdens het uitvoeren van softwareprocessen kunnen worden verkregen en vastgelegd. In dit artikel zal niet worden ingegaan op waarborging van de kwaliteit van de gegevensverkrijging en gegevensvastlegging. Voorbeeldmethoden hiervoor zijn te vinden bij [Grad87], [Pfle95] en [Kitc96].

Tevens kan uit figuur 5 worden afgeleid dat de hoeveelheid ruwe gegevens die tijdens het uitvoeren van softwareprocessen worden vastgelegd, afhankelijk is van de mate van detail en de reikwijdte waarmee de organisatie iets wil weten over het geleverde kwaliteitsniveau. Behalve door het CMM-niveau waarop de organisatie zich bevindt, wordt dit ook bepaald door de invalshoeken die door de belanghebbenden worden gekozen om 'kwaliteit' te kunnen beoordelen.

De binnen het CMM gedefinieerde metrieken zijn op te vatten als een hiërarchisch stelsel en zijn gerelateerd aan de zichtbaarheid en volwassenheid van het softwareproces. Door meerdere hiërarchisch geordende niveaus van metrieken te definiëren, kan elk hoger gelegen CMM-niveau worden gebruikt om steeds meer inzicht in en controle over het softwareproces en -product te krijgen ([Pfle90]). Naarmate het softwareproces volwassener wordt, dienen nieuwe softwaremetrieken te worden toegevoegd. Een dergelijke toevoeging is eveneens nodig als gevolg van nieuw verkregen inzichten dan wel behoeften. Derhalve helpen software(kwaliteits)metrieken een organisatie bij het incrementeel doorlopen van de softwarekwaliteitskringloop om uiteindelijk op CMM-niveau 5 te kunnen uitkomen.

Methode van kwantificeren van de softwarekwaliteitskringloop

Om de kwaliteit van een softwareproces of -product te kunnen kwantificeren, dient de softwareorganisatie een geschikte verzameling metrieken te definiëren. Deze verzameling metrieken dient in overeenstemming te zijn met specifieke behoeften van de organisatie en karakteristieken van de softwareomgeving. De met het uitvoeren van softwaremetingen verkregen meetresultaten worden gebruikt om 'zichtbare kwaliteit' in de betreffende omgeving te kunnen bewaken en beoordelen. Tevens kunnen door het analyseren van de meetresultaten en de hiermee verkregen inzichten verbeteringen worden aangebracht in de betreffende software-entiteiten.

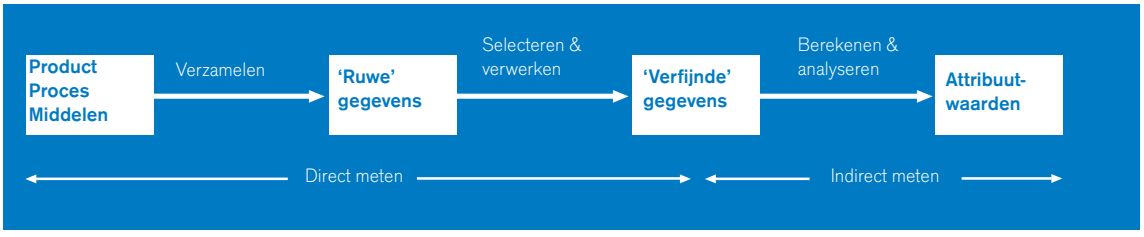
De softwaremeetmethode is toegepast op het correctieve onderhoudsproces van een softwareleverancier. Dit proces is verantwoordelijk voor het herstellen van defecten die door de klanten tijdens het gebruik met het softwareproduct worden geconstateerd.

De van de klanten afkomstige problemen worden door de helpdesk in een database vastgelegd. Volgens beoordeling door het probleembeheer kan een probleem worden veroorzaakt door een defect in het softwareproduct. Dit type problemen wordt vervolgens toegewezen aan het onderhoudsproces van de onderhoudsorganisatie. Op basis van de probleemgegevens wordt het defect (eventueel) opgelost en worden aanvullende gegevens over de aangebrachte wijziging in de database vastgelegd.

Shewart quality cycle

Binnen de CMM-filosofie wordt de 'Shewart quality cycle' als uitgangspunt genomen om (continu) verbeteringen te kunnen aanbrengen in zowel het softwareproces als het -product ([Demi82], [Hump89]). Figuur 6 laat zien dat een organisatie vier opeenvolgende fasen dient te doorlopen om (kwaliteits)verbeteringen aan te kunnen brengen:

1 *Plan*. Beoordelen van de ontwikkelings- en/of onderhoudsomgeving om probleemgebieden te kunnen identificeren en hieruit meetbare primaire doelstellin-



Figuur 5. Verband tussen indirecte en directe softwaremetingen ([Mell92]).

gen te kunnen afleiden. De doelstellingen kunnen met behulp van het 'Goal-Question-Metric' (GQM)-paradigma worden uiteengerafeld in softwarekwaliteitsmetrieken ([Basi88]).

2 *Do*. Beschrijven van het stelsel van softwarekwaliteitsmetrieken.
3 *Check*. Verzamelen van ruwe gegevens van de in het softwareprocesmodel geïdentificeerde entiteiten. Op basis van deze ruwe gegevens kunnen door het toepassen van de metrieken waarden aan attributen van de binnen het model geïdentificeerde entiteiten worden toegekend.
4 *Act*. Analyseren van de meetresultaten ter verkrijging van inzicht in de betreffende software-entiteiten. Op basis van deze bevindingen kunnen verbeteringen worden aangebracht. Tevens kunnen door deze analyses nieuwe probleem- en aandachtsgebieden worden geïdentificeerd.

Het softwaremeetproces is derhalve een systematische methode voor het meten, beoordelen en het (eventueel) aanbrengen van wijzigingen in het softwareproces door gebruik te maken van objectief verkregen gegevens. Dit is dus het toepassen van (geschikte) softwarekwaliteitsmetrieken om kwantitatieve en objectieve beoordelingen te kunnen maken over het geleverde kwaliteitsniveau van het bewuste softwareproces.

Plan: beoordelen softwareomgeving

Het doel van deze stap is om door het beoordelen van de softwareomgeving te komen tot een geschikte verzameling softwarekwaliteitsmetrieken waarmee het kwaliteitsniveau kan worden gemeten. Door het uitvoeren van een CMM-beoordeling kunnen probleem- en aandachtsgebieden worden vastgesteld en kan worden bepaald op welk niveau de organisatie zich bevindt.

Kwantitatief modelleren softwareomgeving

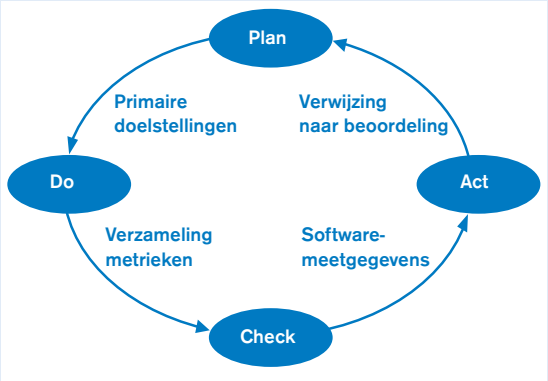
Het beoordelen van de mate van volwassenheid van het softwareproces in kwestie is de eerste stap in het bepalen van de te verzamelen metrieken ([Pfle90]). De beoordeling bestaat uit het in meetbare termen modelleren van de softwareomgeving ([Pfle93]). Het resultaat hiervan is een beschrijving van de software-entiteiten waarover de organisatie iets wil weten; het softwareproces, de softwareproducten en de middelen die nodig zijn om het proces uit te kunnen voeren.

Formuleren primaire doelstellingen

Softwaremetingen dienen te worden voorafgegaan door het formuleren van primaire doelstellingen ([Basi88], [Fent94], [Kitc96]). Door het formuleren van een stelsel doelstellingen wordt tegemoetgekomen aan de specifieke behoeften van de organisatie en derhalve aan de nuttigheidseis die onder anderen Humphrey stelt aan metrieken ([Hump89]).

Basili geeft hiervoor een aantal richtlijnen in termen van de bedoeling van, of de invalshoek waarmee en de omgeving waarin softwaremetingen worden uitgevoerd ([Basi88]):

* *bedoeling*. De doelstelling van softwaremetingen is het karakteriseren, beoordelen, voorspellen of verbeteren van kwaliteitsattributen van software-entiteiten.



Figuur 6. Shewarts 'Plan-Do-Check-Act cycle' ([Demi82]).

Gegeven de proceskarakteristieken voor CMM-niveau 2 en 3 zullen softwaremetingen op CMM-niveau 2 een 'beoordelend' karakter hebben en op CMM-niveau 3 een 'verbeterend' karakter.

* *invalshoek*. De invalshoek die wordt ingenomen bij het uitvoeren van softwaremetingen die betrekking hebben op het geleverde kwaliteitsniveau. Op CMM-niveau 2 zijn de gebruikers- en de externe procesgeoriënteerde invalshoeken te onderscheiden, terwijl op CMM-niveau 3 tevens de productgerichte invalshoek is te onderscheiden.
* *omgeving*. Een beschrijving van de omgeving waarin de software-entiteiten zich bevinden en waarvan de organisatie iets wil weten door het uitvoeren van softwaremetingen.

Uit bovenstaande richtlijnen kan worden afgeleid dat de te formuleren doelstellingen in overeenstemming moeten zijn met de *volwassenheid* van de softwareorganisatie. Immers, een organisatie kan slechts dat meten wat 'zichtbaar' voor haar is en het is derhalve zinloos om doelstellingen te formuleren die niet gekwantificeerd kunnen worden.

> Een organisatie kan slechts dat meten wat voor haar zichtbaar is.

GQM-paradigma

De softwaremeetmethode gebruikt het GQM-paradigma om de geformuleerde doelstellingen te kunnen relateren aan softwaremetrieken. Dit paradigma stelt dat door het top-down uiteenrafelen van doelstellingen in kwantificeerbare vragen hieraan softwarekwaliteitsmetrieken kunnen worden gerelateerd ([Basi88]).

De te definiëren kwaliteitsmetrieken moeten zijn gerelateerd aan de binnen het softwareprocesmodel geïdentificeerde meetbare software-entiteiten. De kwantificeerbare vragen dienen dus betrekking te hebben op de entiteiten die 'zichtbaar' in een bepaalde softwareomgeving aanwezig zijn. Op die manier kunnen per doelstelling een aantal vragen en de hieraan gerelateerde softwarekwaliteitsmetrieken worden geformuleerd.

Het proces is hier op CMM-niveau 2 gedefinieerd en gekwantificeerd. In de onderhoudsomgeving van de softwareleverancier zijn de entiteiten 'problemen', 'defecten' en 'wijzigingen' te onderkennen. Gegeven de karakteristieken van het onderhoudsproces wordt in tabel 1 een opsomming gegeven van de uitwerking van de GQM-methode, waarbij rekening is gehouden met de volwassenheid van het onderhoudsproces.

Tabel 1. Uitwerking GQM-methode voor het correctieve onderhoudsproces.

Doelstellingen	Vragen	Metrieken
Beoordelen tevredenheid klant	Hoeveel problemen beïnvloeden de klant?	Incoming problem rate Fix quality Defect density Fix backlog
	Hoe lang duurt het voordat een probleem is opgelost? (vergeleken met verwachtingspatroon en SLA)	Fix response time % Delinquent fixes Fix backlog
Beoordelen effectiviteit onderhoudsproces	In welke mate worden problemen opgelost?	Fix quality Fix backlog Defect density Incoming problem rate
Beoordelen efficiëntie onderhoudsproces	Waaraan worden de middelen besteed?	Maintenance staffing Problem type distribution Defect type distribution
	Hoe onderhoudbaar is het softwareproduct?	Defect density Fix quality Backlog management index

De in dit artikel beschreven softwaremeetmethode is toegepast op het correctieve onderhoudsproces van een softwareleverancier. Met behulp van deze methode werd de organisatie in staat gesteld om:

* het proces op meerdere niveaus te modelleren, waarbij elk lager gelegen niveau een verdere concretisering van een hoger gelegen niveau voorstelt,
* kan worden gespecificeerd waar, binnen het model, (niet-)softwareproducten worden gecreëerd dan wel worden gewijzigd;
* kan worden bepaald waar in het softwareprocesmodel metingen moeten worden uitgevoerd, door metrieken in het model te relateren aan attributen van meetbare software-entiteiten.

Met behulp van deze methode is een stelsel softwarekwaliteitsmetrieken gedefinieerd waarmee het door het onderhoudsproces geleverde kwaliteitsniveau kan worden gekwantificeerd. De meetresultaten boden het management inzicht in de door het onderhoudsproces geleverde kwaliteitsniveau; daarnaast zullen er stappen ondernomen worden om een softwareverbeteringsprogramma op te stellen (dat gepaard zal gaan met een softwaremeetprogramma).

In het verlengde hiervan wordt het management van de onderhoudsorganisatie in staat gesteld om met behulp van de verkregen meetresultaten:

1 de effectiviteit en efficiëntie van het correctieve onderhoudsproces kwantitatief te definiëren:
   – *effectief:* de mate waarin het correctieve onderhoud effect heeft: de mate waarin defecten in het softwareproduct worden opgelost,
   – *efficiënt:* de mate van effect of resultaat vanuit de door de onderhoudsorganisatie gebruikte middelen;
2 de onderhoudsverbetering c.q. -verslechtering dan wel achteruitgang in de onderhoudbaarheid van softwareproducten te identificeren en kwantificeren;
3 trends te identificeren om toekomstig onderhoud meer voorspelbaar te maken, zodat gekwantificeerde en betekenisvolle schattingen kunnen worden gemaakt. De met behulp van metrieken over een langere periode verkregen informatie kan als basis dienen om het onderhoudsproces naar CMM-niveau 2 te tillen.

Do: beschrijven softwarekwaliteitsmetrieken

Het doel van deze stap is om de in het softwareprocesmodel geïdentificeerde metrieken te beschrijven. Deze beschrijving bevat onder andere, per metriek, de volgende elementen:

* *Waarom?* Dit gedeelte beschrijft waarom dient te worden gemeten. Het waarom vindt zijn neerslag in de doelstellingen die door de organisatie zijn geformuleerd. Ook kunnen de door de CMM-beoordeling geïdentificeerde aandachtsgebieden reden zijn om metingen te gaan uitvoeren.
* *Wat?* Naast een beschrijving van wat door de metriek wordt gemeten, wordt tevens een definitie van het te meten attribuut gegeven.
* *Waar en wanneer?* Door het beschrijven van een softwareprocesmodel kan worden aangegeven waar in het softwareproces metingen dienen te (kunnen) worden uitgevoerd. Tevens zullen sommige metingen een eenmalig karakter hebben, terwijl andere metingen een terugkerend karakter zullen hebben. Een en ander zal afhankelijk zijn van de doelstellingen, behoeften en karakteristieken van de softwareomgeving en zal in het meetprogramma expliciet moeten worden gemaakt.
* *Hoe?* Het hoe-gedeelte handelt over de manier waarop de metriek gemeten dient te worden, welke gegevens benodigd zijn om de metriek te kunnen berekenen en het tool waarmee de metingen uitgevoerd zullen worden.
* *Wie?* In het plan dient naar voren te komen welke typen analyses worden uitgevoerd, door wie de analyses zullen worden gedaan en hoe de analyses kunnen worden gebruikt in het besluitvormingsproces van de betrokken actoren.

De tijdens de planfase geïdentificeerde metrieken zijn in deze fase uitgewerkt. Het resultaat is een softwaremeetplan waarin, naast organisatiespecifieke aandachtsgebieden, de betreffende softwarekwaliteitsmetrieken zijn uitgewerkt.

Check: berekenen softwarekwaliteitsmetrieken

Om het geleverde kwaliteitsniveau te kunnen bewaken en te beoordelen dient de organisatie softwaremetingen uit te voeren. De organisatie maakt hierbij gebruik van de in de subparagraaf 'Do' beschreven verzameling metrieken.

Act: beoordelen en terugkoppelen van de meetresultaten

Op basis van de verkregen meetresultaten kunnen aanbevelingen worden gedaan met betrekking tot het verbeteren van het softwareproces. In het kader van dit artikel voert het te ver om hier uitgebreid op in te gaan. Voorbeelden hiervan zijn terug te vinden bij [Grad92] en [Kitc96].

Conclusie

Organisaties die softwareproducten ontwikkelen en/of onderhouden dienen kwaliteitssystemen te ontwerpen en in te richten die kwalitatief hoogwaardige softwareproducten kunnen voortbrengen. Uit de literatuur over softwareontwikkeling kan worden opgemaakt dat de jaren negentig zich laten kenmerken door de bewustwording in de software-industrie dat continue kwaliteitsverbetering van doorslaggevende betekenis is dan wel kan zijn om op de langere termijn te kunnen overleven.
Voor deze organisaties zijn kwaliteitssystemen beschreven, die hen ondersteunen in het continu kunnen doorvoeren van verbeteringen in hun softwareprocessen en -producten. Het Capability Maturity Model van Humphrey en Paulk is wellicht het bekendste.

Binnen het CMM wordt de 'Shewart quality cycle' als uitgangspunt genomen om (continu) kwaliteitsverbeteringen te kunnen aanbrengen in zowel het softwareproces als -product. Tijdens het incrementeel doorlopen van de kwaliteitskringloop kunnen softwaremetrieken de organisatie in kwestie helpen om (continu) verbeteringen door te voeren; verbeteringen dienen gepaard te gaan met softwaremetingen. In de softwareliteratuur wordt veel aandacht besteed aan softwaremeetmethoden, softwaremetrieken en softwaremetingen.

> Voorkomen moet worden dat men software-attributen meet terwijl de verkregen meetresultaten voor niemand van belang zijn.

De in dit artikel beschreven softwaremeetmethode maakt gebruik van Shewarts kwaliteitskringloop, Basili's GQM-paradigma, Humphreys CMM en Fentons softwareprocesmodelleermethode.

In tegenstelling tot de in de softwareliteratuur beschreven meetmethoden gebruikt de in dit artikel beschreven benadering een softwareprocesmodel en laat de methode verschillende (kwaliteits)invalshoeken toe. Hiermee wordt tegemoetgekomen aan de nuttigheidseis van de verzameling metrieken omdat ze in een praktische behoefte kunnen voorzien. Dit is van belang omdat vele software-attributen kunnen worden gemeten, terwijl de verkregen meetresultaten in sommige gevallen voor niemand van belang zijn. Tevens wordt de organisatie gedwongen om door het modelleren van het softwareproces na te denken over de componenten waaruit het proces is opgebouwd, zodat kan worden bepaald waar (binnen het model) softwaremetrieken kunnen worden gedefinieerd.

Samenvattend komt de beschreven softwaremeetmethode tegemoet aan de in de softwareliteratuur beschreven bevindingen:

* Softwaremetingen moeten zijn gerelateerd aan de karakteristieken van de softwareomgeving en de behoeften van actoren binnen deze omgeving.
* De initiële verzameling kwaliteitsmetrieken waarmee softwaremetingen kunnen worden uitgevoerd, dient in beginsel 'klein' te worden opgezet; naarmate het soft-

wareproces volwassener wordt, kunnen additionele metrieken aan deze verzameling worden toegevoegd.
* Softwaremetingen dienen te zijn gerelateerd aan de *zichtbaarheid* en *volwassenheid* van een softwareproces.
* Softwaremetingen maken een noodzakelijk onderdeel uit van kwaliteitsverbeteringen.

Literatuur

[Basi88]
V.R. Basili and H.D. Rombach, *IEEE Transactions on Software Engineering: The TAME project: Towards improvement-oriented software environments*, IEEE Software, June 1988, pp. 758-773.

[Bhid90]
S. Bhide, *Generalized software process-integrated metrics framework*, Journal of Systems and Software, December 1990, pp. 249-254.

[Boeh78]
B.W. Boehm, J.R. Brown, H. Kaspar, M. Lipow, G.J. MacLeod and M.J. Merritt, *Characteristics of software quality*, TRW Series of Software Technology, Vol. 1, North Holland Publishing Company, Amsterdam/New York/Oxford 1978.

[Boeh81]
B.W. Boehm, *Software engineering economics*, Prentice-Hall, Englewood Cliffs, New Jersey 1981.

[Boeh84]
B.W. Boehm, *Verifying and validating software requirements and design specifications*, IEEE Software, January 1984, pp. 75-88.

[Bush90]
M.E. Bush and N.E. Fenton, *Software measurement: A conceptual framework*, Journal of Systems and Software, December 1990, pp. 223-231.

[Cros79]
P.B. Crosby, *Quality is Free: the art of making quality certain*, McGraw-Hill, New York 1979.

[Dele90]
G.P.A.J. Delen en D.B.B. Rijsenbrij, *Kwaliteitsattributen van automatiseringsprojecten en informatiesystemen*, Informatie, januari 1990, pp. 46-55.

[Demi82]
W.E. Deming, *Out of the Crisis*, MIT Center for Advanced Engineering Study, Cambridge (MA) 1982.

[Evan95]
M.M. Evanco, *Modelling the effort to correct faults*, Journal of Systems and Software, April 1995, pp. 239-249.

[Fent91]
N.E. Fenton, *Software metrics: a rigourous approach*, Chapman & Hall, London 1991.

[Fent94]
N.E. Fenton, *Software measurement: A necessary scientific basis*, IEEE Transactions on Software Engineering, March 1994, pp. 199-206.

[Garv84]
D. Garvin, *What does product quality really mean?*, Sloan Management Review, Fall 1984, pp. 25-43.

[Gilb88]
T. Gilb and S. Finzi, *Principles of software engineering management*, Addison-Wesley, Reading (Mass.), 1988.

[Grad87]
R.B. Grady and D.L. Caswell, *Software metrics: Establishing a company-wide program*, Prentice Hall, Englewood Cliffs, New Jersey 1987.

[Grad92]
R.B. Grady, *Practical software metrics for project management and process improvement*, Prentice-Hall, Englewood Cliffs, New Jersey 1992.

[Grem84]
L.L. Gremillion, *Determinants of program repair maintenance requirements*, Communications of the ACM, August 1984, pp. 826-832.

[Henr96]
J. Henry, R. Blasewitz and D. Kettinger, *Research and Practice: Defining and implementing a measurement-based software maintenance process*, Journal of Software Maintenance, February 1990, pp. 79-100.

[Hump89]
W.S. Humphrey, *Managing the Software Process*, SEI Series in Software Engineering, Addison-Wesley Publishing Company, Reading (Mass.) 1989.

[ISO92]
ISO 9126, *Information Technology: software product evaluation: quality characteristics and guidelines for their use*, International Organisation for Standardization, Geneva 1992.

[ISO94]
ISO 8402, *Quality management and quality assurance: vocabulary*, International Organisation for Standardization, Geneva, 2nd edition, 1994.

[Jura70]
J.M. Juran and F.M. Gryna, *Quality planning and analysis: From product to development through use*, McGraw-Hill, New York 1970.

[Kan95]
S.H. Kan, *Metrics and models in software quality engineering*, Addison-Wesley Publishing Company, Menlo Park, California 1995.

[Kitc96]
B. Kitchenham, *Software metrics: Measurement for software process improvement*, NCC Blackwell, Cambridge, Massachusetts, USA, 1996.

[Lehm80]
M.M. Lehman, *On understanding laws, evolution and conservation in the large program lifecycle*, Journal of Systems and Software, January 1980, pp. 213-221.

[Lien81]
B.P. Lientz and B.E. Swanson, *Problems in application software maintenance*, Communications of the ACM, November 1981, pp. 763-769.

[Looij95]
M. Looijen, *Beheer van informatiesystemen*, Kluwer Bedrijfswetenschappen, Deventer 1995.

[McCa77]
J.A. McCall, P.K. Richards and G.F. Walters, *Factors in software quality*, Rome Air Development Center, November 1977.

[Mell92]
P. Mellor, *Failures, faults and changes in dependability measurement*, Journal of Information and Software Technology, October 1992, pp. 640-652.

[Neil90]
M. Neil, R.J. Cole and D. Slater, *Research and Practice: Measures for maintenance management: a case study*, Journal of Software Maintenance, February 1990, pp. 223-240.

[Paul93]
M.C. Paulk, B. Curtis, M.B. Chrissis and C.V. Weber, *Capability Maturity Model for Software, Version 1.1*, IEEE Software, July 1993, pp. 18-27.

[Pfle90]
S.L. Pfleeger and C. McGowan, *Software Metrics in the Process Maturity Framework*, Journal of Systems and Software, December 1990, pp. 255-261.

[Pfle93]
S.L. Pfleeger, *Lessons learned in building a corporate metrics program*, IEEE Software, May 1993, pp. 67-74.

[Pfle95]
S.L. Pfleeger, *Maturity, models and goals: How to build a metrics plan*, Journal of Systems and Software, February 1995, pp. 143-155.

[Roch94]
J.M. Roche, *Software metrics and measurements principles*, Software Engineering Notes, January 1994, pp. 77-85.

[Sass96]
H. Sassenburg, G. Matser en P. Kazil, *Software Process Improvement: Waarom en wanneer?*, Informatie, juli/augustus 1996, pp. 50-57.

[Trie94]
J. Trienekens, *Tijd voor kwaliteit, werken aan betere informatiesystemen*, Thesis Publishers, Amsterdam 1994.

[Valla88]
S.R. Vallabhaneni, *Auditing the maintenance of software*, Prentice-Hall, Englewood Cliffs, New Jersey 1988.