# Kwaliteit op tijd in productontwikkeling

## Aloude kwaliteitsprincipes maken ontwikkelproces voorspelbaar

*Een nieuw product ontwikkelen duurt altijd langer dan je denkt. Als 90% gereed is, begin je aan de volgende 90%. Hoe komt dat? En wat kunnen we door aan doen? Door het toepassen van aloude kwaliteitsprincipes op alle facetten van het ontwikkelproces, wordt dat proces voorspelbaar en beheersbaar. Overigens zonder dat de creativiteit van de ontwikkelaars onnodig wordt ingeperkt. Integendeel: ze krijgen nu optimaal de tijd om aan werkelijk creatieve zaken te werken!*

Ir. N.R. Malotaux, N R Malotaux - Consultancy

Als we ontwikkelaars leren volgens de juiste methoden te werken, blijken ze goed in staat nauwkeurige tijdinschattingen te maken en binnen die tijd foutloze producten te realiseren. Het grootste probleem is echter psychologisch: ontwikkelaars en hun management lijken het uitlopen van ontwikkeltijden en het achteraf repareren van onvolkomenheden (debuggen) als iets onontkoombaars te hebben aanvaard.

### Kwaliteit

De enig bruikbare (want verifiëerbare) definitie van kwaliteit is: 'Voldoen aan de afgesproken eisen'; het product doet precies wat het moet doen. Voordat we een kwaliteitsproduct kunnen realiseren, moeten we dus eerst exact beschrijven wat het moet doen.

Als de eisen niet goed worden afgesproken is het onmogelijk het juiste product te maken. Men kan dan immers niets controleren. Ontwikkelaars hanteren vaak de smoes dat hun klant niet goed kan bepalen wat hij wil en dat hij dus maar moet afwachten wat hij krijgt: de ontwikkelaars gaan gewoon maar iets maken. Zo'n trial-and-error-methode is niet meer acceptabel (tenzij doelbewust uitgevoerd volgens de 'spiraalmethode' - Boehm 88). En als de expertise ontbreekt om te beoordelen wat er moet worden gemaakt, dan moet die expertise worden ingehuurd.

### Op tijd

Wat betekent 'op tijd'?

- gisteren?
- vóór de volgende beurs?
- time-to-market?
- binnen de berekende doorlooptijd?

Het enig juiste antwoord is *binnen de berekende doorlooptijd*. Een andere aanname is zinloos en werkt zelfs averechts. Dit kunnen we illustreren met figuur 1. Als je tien ontwikkelteams dezelfde opdracht geeft, zal de per team benodigde realisatietijd de in de figuur geschetste kansverdeling opleveren. Niet elk team presteert even snel en sommige raken nooit klaar. Wel is duidelijk dat er een minimale ontwikkeitijd is. Het lukt niemand om in een kortere tijd de ontwikkeling af te krijgen. Als je een gemiddeld ontwikkelteam vraagt hoe lang de ontwikkeling gaat duren, zul je zien dat men meestal een optimistische kijk heeft op de benodigde tijd (stippellijn 'schatting'). Komen ze met deze tijd bij de baas of bij de klant, dan krijgen ze meestal te horen dat deze tijd te lang is. Wanneer ze vervolgens toch worden gedwongen om die tijd te accepteren, zijn ze na die tijd natuurlijk nog niet klaar.

### Onrealistische realisatietijd

Ze *konden* ook niet klaar zijn. Door het gemopper van de baas en de klant worden ze echter wel opgezadeld met het gevoel dat ze gefaald hebben. In ieder geval worden ze niet gewaardeerd. Dit is slecht voor de motivatie. Als men zich realiseert dat *motivatie de motor is van productiviteit*, is het duidelijk dat een gemiddeld team daardoor niet in de gemiddelde tijd klaar is, maar nog later. Kortom, het dicteren van een onrealistische realisatietijd werkt averechts: het product komt alleen maar nog later.

Wat leren we hiervan? Moeten we dan maar afwachten tot ze eindelijk klaar zijn? Nee. Enige druk scheelt wel degelijk. Maar deze druk moet van binnenuit komen. Als ontwikkelaars leren de benodigde tijd beter in te schatten, kunnen ze realistische schattingen afgeven, waar ze vervolgens ook voor instaan: er ontstaat commitment. Dat wil zeggen dat de baas en de ontwikkelaar een afspraak hebben, waarvan men weet dat hij haalbaar is. De baas weet dat hij er op kan rekenen en de ontwikkelaar voelt zich moreel verplicht, is

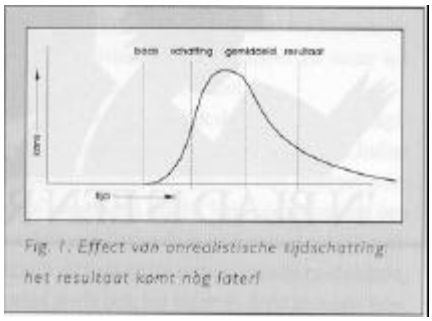



Fig. 1. Effect van onrealistische tijdschatting: het resultaat komt nóg later!

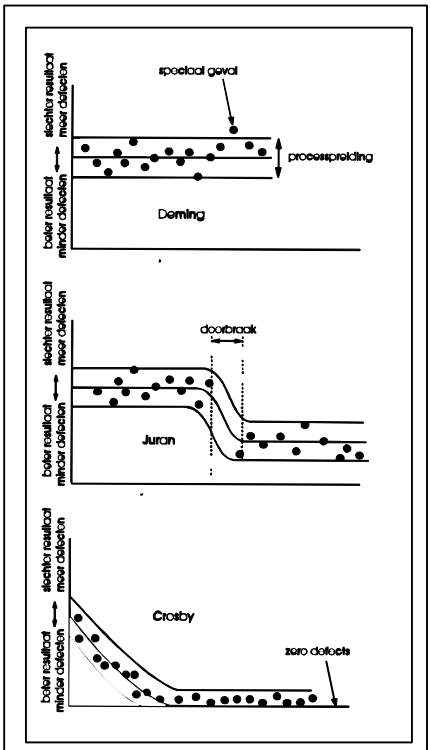



*Fig. 2. De inzichten die leiden tot processen die een juist functionerend product, op tijd klaar, op moeten leveren, stoelen op aloude kwalitetsprincipes*

daartoe ook in staat en zorgt er gewoon voor deze tijd te halen.
Is de benodigde tijd (vaak om zeer valide redenen) onacceptabel, dan volstaat niet het verkorten van de tijd, maar moet men de eisen bijstellen of meer mensen inzetten. Vergeet daarbij echter niet de Wet van Brooks: 'Adding more people to a late project makes it later' (Brooks, 1975). Dat wil zeggen dat het inzetten van meer mensen alleen zin heeft als dit goed wordt gepland en niet als antwoord op een paniekssituatie. Werken volgens de verderop geschetste first-time-right methoden levert overigens al een significante besparing in ontwikkeltijd c.q. doorlooptijd, zonder dat extra mensen hoeven te worden ingezet.
Het is dus belangrijk dat ontwikkelaars hun benodigde tijd beter leren schatten.
Time-management is één van de pijlers van het PPP: het 'Personal Performance Proces' (zie kader Personal Software Process). Een andere belangrijke pijler is *defect-management,* zie verder). Via deze methode kunnen ontwikkelaars binnen één jaar de onnauwkeurigheid in hun tijdschatting verbeteren van -0 ... +π naar ±20%. En wanneer ze die schattingen per subproject doen, middelt de onnauwkeurigheid over het totale project uit tot een zeer handelbaar niveau. Omdat de schatting nu realistisch is, is die verdedigbaar, bespreekbaar en haalbaar, en is commitment geen probleem. Op tijd is hiermee voor iedereen een duidelijk begrip geworden.

## Ontwikkelprojecten

Bij het uitvoeren van ontwikkelprojecten dient men oog te hebben voor drie P's: het *Product,* het *Project* en het *Proces*.

**Het product: het programma van eisen**

Om goed te definiëren wat er moet worden ontwikkeld, moet eerst een *programma van eisen* (PvE) worden bepaald. Dit is een document waarin eerst de atmosfeer wordt beschreven waarin het te ontwikkelen product moet ontstaan en gedijen, gevolgd door doelstellingen, normen, uitgangspunten, toepassingsgebieden, gebruikersgroepen en verdere eisen. Het PvE moet worden gecontroleerd op de volgende hoofdpunten:

- relevantie;
- compleetheid;
- consistentie;
- verifieerbaarheid.

Het opnemen van de methode van validatie *bij iedere eis* geeft enerzijds extra interpretatie aan de eis, terwijl tevens wordt gedefiniëerd wanneer aan de eis is voldaan, hetgeen de voorwaarde voor kwaliteit is.
Het PvE bepaalt wat, waarom en voor wie moet worden ontwikkeld. Het is een leidraad voor wat er verder moet gebeuren. Elke onduidelijkheid of onjuistheid resulteert in problemen in het product en moet met alle middelen worden voorkomen. Het PvE moet compleet en afgerond zijn voordat met de volgende projectfasen kan worden begonnen. Dat er tijdens de verdere ontwikkeling wijzigingen kunnen worden aangebracht is mogelijk, maar moet zoveel mogelijk worden ontmoedigd. Wijzigingen in het PvE veroorzaken per define defecten, met alle desastreuze gevolgen van dien, zoals we hierna zullen aan tonen (zie ook kader Defecten). Defecten die worden gevonden moeten altijd worden geanalyseerd met de vragen 'Hoe kon dit optreden?' en 'Hoe kunnen wij zorgen dat dit soort defect in de toekomst niet meer optreedt?'. Defecten veroorzaakt door wijzigingen in het PvE dienen evenzo te worden geanalyseerd om te zien hoe het PvE-generatieproces kan worden verbeterd om het aantal wijzigingen gedurende een ontwikkeling te minimaliseren.

**Het product: conceptontwikkeling**

Na het opstellen van het programma van eisen volgt de conceptontwikkelingsfase:
beschikbare technieken en oplossingsmogelijkheden worden verzameld, vergeleken en gekozen. Deze fase heeft twee polen:

- weten wat er moet worden gerealiseerd (beschreven in het PvE);
- kennis over de eigenschappen, mogelijkheden en beperkingen van technieken en middelen die ter beschikking staan om deze eisen te realiseren.

Het is aan de projectleiding om te herkennen welke kennis moet worden aangeboord, waar deze te vinden is en hoe ze kan worden ingezet. Daartoe kunnen we bij de conceptontwikkeling twee fasen onderscheiden, die overigens wel wisselwerking vertonen:

- Algemene systeemconcepten. Die werken we eerst zelf uit: wat zijn de algemene oplossingsmogelijkheden voor de gestelde eisen? Op welke deelgebieden moeten we wat nog verder onderzoeken en uitwerken?

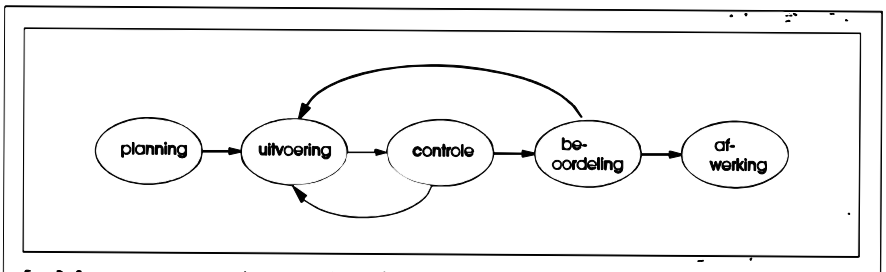


Fig. 3. Structuur van een subproces volgens first-time-right-methode.

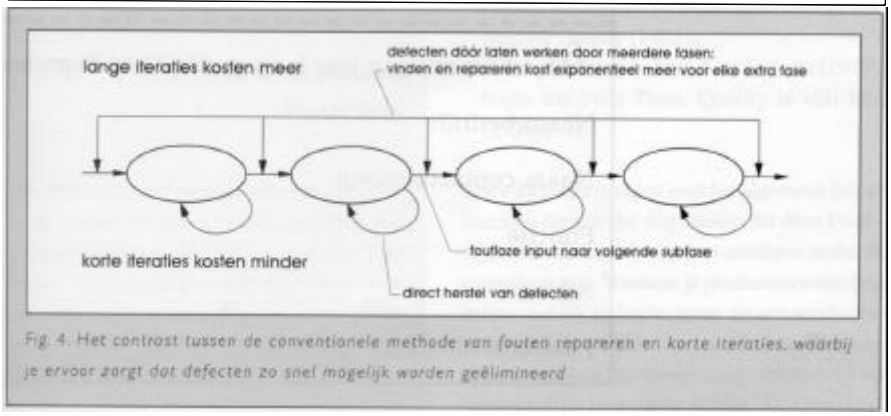


Fig. 4. Het contrast tussen de conventionele methode van fouten repareren en korte iteraties, waarbij je ervoor zorgt dat defecten zo snel mogelijk worden geëlimineerd

### Projectleidercategorieën (volgens Malotaux):

**Cat 1.** Er is helemaal geen projectleider. Het project dobbert stuurloos rond.
**Cat 2.** Er is een projectleider aangewezen, maar die weet niet dat hij het is, of anderen weten niet wie het is, of niemand realiseert zich wat dat betekent.
**Cat 3.** De projectleider is een projectvolger: hij volgt het project, ziet hoe het uit de hand loopt en rent zich vervolgens rot. Hij hoopt steeds dat ooit alles vanzelf beter gaat en op zijn pootjes terechtkomt: hij struisvogelt.
**Cat 4.** Dit is de echte projectleider. De dirigent voor het orkest, de kapitein op het schip. Hij krijgt de ruimte als projectleider. Hij ontwikkelt visie, strategieën en scenario's. Hij maakt een realistisch plan, gebaseerd op ervaring, historische kennis en -resultaten en zorgt vervolgens dat het op de juiste wijze gebeurt. Direct goed, op tijd klaar en met beheerste kwaliteit.


Artikel zoals verschenen in ELEKTRONICA nr 6, juni 1999

- Specifieke deelconcepten. In het geval van toeleveranciers van ontwikkeling, maar ook van productie, kan het nodig zijn ook deze op een gegeven moment reeds te betrekken in de conceptontwikkeling.

Uiteindelijk wordt een specificatie gegenereerd, op basis waarvan onvoorwaardelijk duidelijk is wat moet worden gerealiseerd. Elk onderdeel van deze specificatie is terug te voeren op documentatie die tijdens de PvE- en conceptontwikkelingsfasen is gegenereerd.

**Het product: uitvoering**

De bedoeling is dat de nu afgeronde documenten de basis vormen voor een uitvoeringsproces, waar zich eigenlijk geen verassingen meer voordoen. Alle onzekerheden zijn overdacht en eventueel in experimenten c.q. miniprojecten uitgezocht. Elke vraag die nog open stond is beantwoord. Kortom de nu volgende fasen kunnen 'direct goed' (first time right) worden uitgevoerd. Het resultaat is dan een product dat precies doet wat het moet doen en op tijd klaar is.

## Het project

Om ontwikkelprojecten op tijd te laten eindigen moet men, zoals boven beschreven, zich eerst goed realiseren wat 'op tijd' betekent. Een project zal echter nooit goed (juiste kwaliteit, op tijd) lopen wanneer er geen projectleider van de vierde categorie aan het roer staat (zie kader Projectleidercategorieën). Dit is een projectleider die bijvoorbeeld direct actie onderneemt bij afwijkingen in de planning, en vervolgens analyseert hoe de vertraging heeft kunnen optreden en deze kennis gebruikt om bij volgende projectonderdelen te voorkomen dat een dergelijke vertraging nog eens optreedt. Het uitlopen van de projectduur gebeurt meestal sluipend (How do projects get late? Day by day!). Het komt overigens bij de meeste projecten voor en het is één van de redenen waarom ik mij met deze materie ben gaan bezighouden, met als doel om hierin significante en blijvende verbetering te brengen. De meeste projecten duren langer dan gepland. Er is een mentaliteitswijziging,

**Personal Software Process**

Het PPP heeft veel gemeen met het PSP (Personal Software Process, Humphrey, 1995). Het door ons gehanteerde PPP is echter veel algemener toepasbaar gemaakt dan alleen voor software, en is realistischer aangepast aan de werkelijkheid dan het vaak al te rigoreurs bevonden PSP.

gecombineerd met de juiste methode voor nodig om te zorgen dat projecten routinematig wèl op tijd, zonder stress en met het gewenste resultaat worden afgerond. Dit wordt bij de opleiding niet aangeleerd en is dus niet direct aan te rekenen. Er is echter wel iets aan te doen.

## Het proces

Om te voorkomen dat bij iedere nieuwe ontwikkeling alle fouten opnieuw worden gemaakt, dient men zich goed te realiseren waar men eigenlijk mee bezig is. Hoe voer je zo goed mogelijk een productontwikkeling uit en hoe run je het beste een project. Men moet zich dus ook bezighouden met bet productontwikkelings-*proces* en het project-*proces*. Daarnaast moet men een onderscheid maken tussen management-processen (voor de performance als organisatie) en individuele performance-processen (PPP - Personal Performance Process). Het management volgt processen voor het opzetten en coördineren van ontwikkelingen, en schept voor de individuele ontwikkelaars een optimale atmosfeer. Bij de meeste uiteenzettingen over dit onderwerp blijft het daarbij. Minstens zo belangrijk zijn echter de individuele performance processen, waarmee de ontwikkelaars immers het eigenlijke werk optimaal moeten kunnen uitvoeren. Door deze processen in kaart te brengen, te volgen en de effectiviteit te analyseren kan men werken aan de verbetering ervan. Als men uit ervaring twijfelt aan de mogelijkheid om in één keer, en nog wel op tijd, de juiste producten te kunnen realiseren, zijn de gehanteerde processen blijkbaar niet goed bekend en in ieder geval niet geoptimaliseerd.

**Het proces: kwaliteitsprincipes**

De inzichten die leiden tot processen die een juist functionerend product, op tijd klaar, op moeten leveren, stoelen op aloude kwaliteitsprincipes, zoals die zijn verkondigd door lieden zoals:

- Deming (bracht kwaliteitsideedn naar Japan (1948), Out of the crisis (1986));
- Juran (bracht nog meer kwaliteitsideeën naar Japan (1950), Quality Control Handbook (1951), Managerial Breakthrough (1964), Planning for Quality (1988));
- Crosby (Quality is free, Zero Defects (1979), Right the First Time, Quality is still free (1996)).

Ontwikkelaars dachten over het algemeen (en de meesten denken dat nog steeds) dat deze kwaliteitsmethodieken niet op hun creatieve ambacht toepasbaar zijn. Wanneer je productontwikkeling echter niet als ambacht, maar als een productieproces bekijkt, blijken opeens alle bekende technieken voor kwaliteitsbeheersing wel degelijk op ontwikkeling toepasbaar te zijn.

We ontwikkelen weliswaar steeds iets (meestal niet geheel) nieuws, maar de *wijze* van ontwikkelen is in essentie steeds hetzelfde. Daarmee *produceren* we ontwikkelingen.

Heel schematisch aangeduid kunnen we stellen (zie figuur 2) dat Deming aangaf dat uitvoerenden door het proces worden geleid en dat ze individueel maar een zeer kleine invloed op het procesresultaat hebben. Juran legde de nadruk op de (door Deming overigens ook al aangegeven) noodzaak van 'managerial breakthrough': het optimaliseren van het proces om tot betere resultaten te komen. Crosby wees de weg naar een rigoureuze aanpak van niet stapsgewijs, maar continu het proces op alle mogelijke punten aanpakken, zodat het procesgemiddelde op zero-defects komt te liggen. Deze laatste benadering is een duidelijke doorbraak in het kwaliteitsdenken: we gaan het proces niet in de eerste plaats optimaliseren voor een beheerste hoeveelheid defecten, maar we gaan voor zero-defects: foutloze resultaten. Financiële afdelingen werken altijd al volgens dit principe, omdat dat nu eenmaal van ze wordt verwacht: boekhouders leren vanaf het begin dat fouten maken een doodzonde is en ze leren methoden om fouten te detecteren en te elimineren. Waarom kunnen productontwikkelaars dat dan niet? Omdat dat niet van begin af aan van ze wordt verwacht.

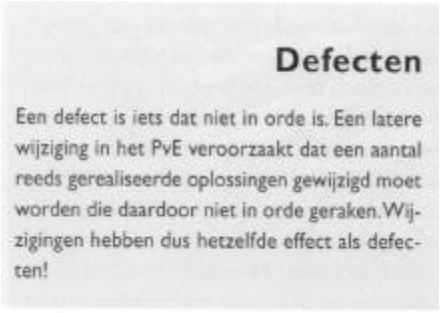
Een foutloos product is het enige dat je als klant doodnormaal zou vinden. En first-time-right-methoden zijn er om dat mogelijk te maken.

**Het proces: first-time-right**

First-time-right, of hoe doen we alles direct goed en hoe optimaliseren we dat proces? De noodzaak van de weg daartoe illustreren we met de volgende twee stappen:

**1.** Fouten maken is menselijk. Als je iemand vraagt of hij alles in één keer goed kan doen, dan zal hij meestal antwoorden dat mensen nu eenmaal fouten maken. Vervolgens trekt men daaruit echter niet de voor de hand liggende consequentie: als we iets hebben gepresteerd, dan zitten er dus fouten in. Fouten maken is immers menselijk. Als we in de loop van het ontwerpproces iets *denken* af te hebben, dan *weten* we nu dus dat er *nog fouten in zitten.*

**2.** Hoe langer een fout in het systeem blijft, des te kostbaarder is de reparatie. De kosten (tijd/geld) om een fout te herstellen zijn exponentiëel (1x, 10x, 100x, 1000x, 10000x, ...) hoger, naarmate de fout langer in het systeem blijft. Denk bijvoorbeeld aan de Pentium-bug, die veroorzaakt werd door een foutje in een algoritme voor het vertalen van een formule naar een tabel, waarbij één entry in de tabel verkeerd werd berekend. En het geval met de Ariane V ligt ook nog vers in het geheugen.




Wanneer moeten we dus de geï ntroduceerd fouten opsporen?

Aan het eind van het ontwerpproces?

- testen;
- debuggen.

Direct na het introduceren van de fout?

- controle;
- inspectie.

Met testen kunnen we - als we fouten vinden - wel aantonen dat er fouten in een systeem zitten. We kunnen echter nooit bewijzen dat een systeem foutloos is, daarvoor zijn de producten die wij ontwerpen te complex. Testen is dus altijd een steekproef, die in de praktijk vaak nog random, ongeorganiseerd, wordt uitgevoerd. Het enige doel van een test mag zijn: constateren *dat het werkt,* nooit het vinden van defecten.

Debuggen is een woord dat uit het woordenboek moet worden geschrapt: het is een vrijbrief voor slordig werken. Uit uitgebreid onderzoek (vooral op softwaregebied) is gebleken dat het defectoplossend vermogen van testen en debuggen dermate bedroevend is (er worden veel te veel defecten niet gevonden en ook teveel defecten toegevoegd) dat deze methoden inadequaat zijn voor het leveren van kwaliteitsproducten.

Nee, als we eenmaal weten dat we, menselijk als we zijn, defecten introduceren, die naarmate ze later worden gevonden steeds kostbaarder zijn om te vinden en te elimineren, dan is het noodzakelijk ons proces in te richten op het *direct na het introduceren* elimineren ervan. De methoden daartoe zijn (zie figuur 3) controle (door de uitvoerder zelf) en inspectie/beoordeling (door collega's).

Controles en inspecties worden niet random uitgevoerd, maar aan de hand van checklists en volgens een goed opgezet programma. Resultaat: circa 90 - 95% reductie van defecten (omdat mensen nu eenmaal fouten maken, is het detectieproces ook niet perfect). Het betekent wel dat het aantal defecten is gedecimeerd, betgeen een belangrijke tijdwinst oplevert.

De behaalde winst uit zich op een aantal fronten:

- het kost minder om alles direct goed (menselijker: zo goed mogelijk) te doen
- direct goed doen leidt tot een beter resultaat**:** reparaties laten altijd littekens achter en vergroten het risico van problemen in het veld, die nog veel kostbaarder zijn om op te lossen;
- problemen in het veld kosten kostbare tijd die beter aan nieuwe ontwikkelingen kan worden besteed;
- elke dag eerder op de markt levert een dag meer winst op;
- elke dag later dan de concurrent levert nog minder winst. Time-to-market is geen loze kreet maar een economische realiteit.

Het komt er op neer dat je niet zo gauw 'teveel' tijd besteedt aan het goed voorbereiden en goed controleren van wat je gedaan hebt. Dat lijkt een kostbare investering vergeleken met de oorspronkelijke planning, die uitgaat van 'als alles goed gaat'. Maar alles gaat niet goed: we weten nu immers dat we fouten maken. In de meeste organisaties is echter helemaal geen capaciteit gepland voor de controlefase en nog minder voor inspectie door collega's. Daardoor heeft men uiteindelijk meer tijd nodig voor het achteraf repareren van defecten.

En daardoor is er zeker geen tijd voor controles en inspecties. Men zit dus in een vicieuze cirkel die zonder doelbewuste actie niet wordt doorbroken. In figuur 4 is het contrast aangegeven tussen de conventionele methode van fouten repareren wanneer ze toevallig gevonden worden en de korte iteraties, waarbij je ervoor zorgt dat defecten zo snel mogelijk worden geëlimineerd.

## Conclusie

Routinematig het juiste product op tijd realiseren is mogelijk. Er is niet één eenvoudig recept dat je kunt volgen om dit te bereiken. 'There is no silver bullet' zei Brooks al (Brooks, 1987).

Belangrijke pijlers zijn:

- Eerst denken en dan pas doen. Reparaties in denken (iteraties) zijn veel goedkoper dan later. Dus eerst bepalen wat er moet gebeuren (PvE, specificatie) en dan pas uitvoeren. Dit kan beslist veel rigoureuzer dan het nu gebeurt.
- Projectmanagement en -discipline.
- Procesbeheersing en -optimalisatie.
- Time-management. Leer hoeveel tijd bepaalde werkzaamheden kosten en daarmee de schattingen van nieuw werk optimaliseren.
- Defectmanagement. Mensen maken fouten. Elimineer ze direct. En leer ervan.

Ga voor zero-defects! Alleen al door het omzetten van die schakelaar doe je een grote stap vooruit.

*Inl.: Meer stof over het beschreven onderwerp, alsmede een uitgebreide literatuurlijst is te vinden op www.malotaux.nl/nrm.*

***N R Malotaux - Consultancy,***
***Ir. N.R. Malotaux, tel.: (030) 228 88 68,***
***e-mail: niels@malotaux.nl***
***web: www.malotaux.nl/nrm***





Artikel zoals verschenen in ELEKTRONICA nr 6, juni 1999