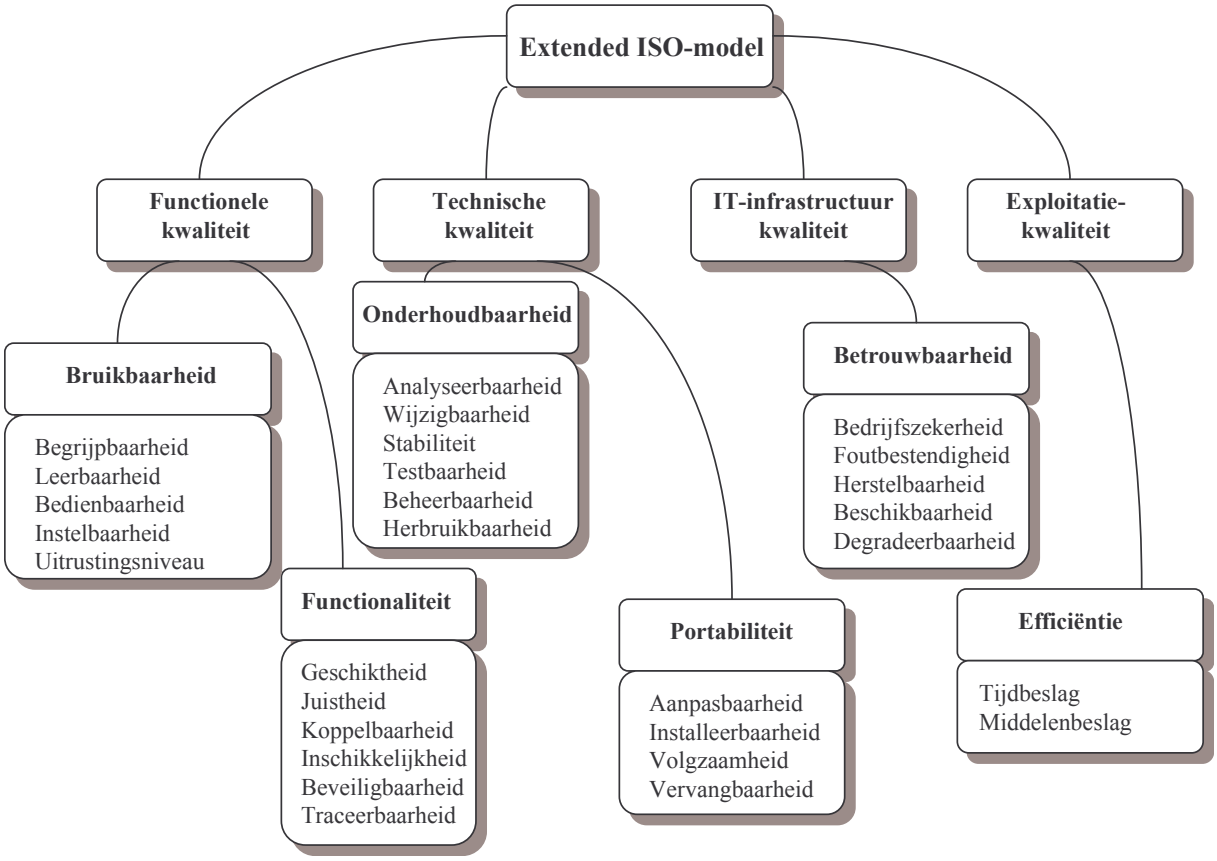
ITIL en de SoftwareLifeCycle (vervolg)

Door ing. B. de Best

In het vorige artikel hebben we stil gestaan bij de relatie tussen ITIL en de fasen van de SoftwareLifeCycle. We hebben gezien dat zowel vanuit beheer en exploitatie als vanuit systeemontwikkeling en applicatiebeheer een goede afstemming van werkzaamheden gewenst is. Aangetoond is dat hiervoor handig gebruik kan worden gemaakt van de Service Activity Matrix (SAM).

Zoals in het eerste artikel is gesteld gaat dit artikel in op de acceptatiecriteria die per onderkende relatie in de SAM zijn te definiëren aan de hand van het Extended ISO 9126 model. Om dit duidelijk te maken worden eerst de kwaliteitsattributen van het ISO 9126 model weergegeven. Daarna wordt aan de hand van een eenvoudige SAM de relatie tussen ITIL en het extended ISO 9126 model besproken. Tenslotte wordt ingegaan op de uitwerking van zo'n SAM relatie aan de hand van de ISO 9126 acceptatiecriteria.

Het Extended ISO 9126 model
In Figuur 1 zijn de kwaliteitsattributen van het Extended ISO 9126 model weergegeven. Er wordt vanuit 4 gezichtspunten gekeken naar de kwaliteit van applicaties te weten, functioneel, technisch, IT Infrastructureel en exploitatie. Per gezichtpunt zijn twee kwaliteitattribuutgroepen benoemd. Het idee om met 4 gezichtspunten naar de kwaliteitsattributen te kijken is gebaseerd op de filosofie 'Life Cycle Enabling' van Roccade [ROCCADE 1998]. Het voordeel hiervan is dat de toepasbaarheid van kwaliteits attributen duidelijker wordt. De mapping naar de ITIL processen en de verschillende fase van de softeware lifecycle geven hier nog een verdere detaillering aan. De kwaliteitsattributen zelf worden niet besproken omdat hiervoor goede literatuur voorhanden is [RADAR 1996].



Figuur 1, R.B. de Vries, [ROCCADE 1997]

De bedoeling van deze kwaliteitsattributen is om per attribuut een checklijst op te stellen voor een te accepteren applicatie. Dit kan per type applicatie verschillen, een client-server applicatie kan bijvoorbeeld een andere checklijst vereisen dan een java applicatie. Naast variatie per type applicatie zullen ook de business requirements van invloed

kunnen zijn op de invulling van de kwaliteitsattributen. Bij een hoge eis aan ‘time-to-market’ zullen er minder hoge eisen gesteld kunnen worden aan de applicaties. We komen hierbij weer terug op de al-oude balans van stabiliteit en flexibiliteit.

Het Extended ISO 9126 model in relatie met ITIL
Het is interessant om te kijken naar de relatie tussen de kwaliteitsattributen en de ITIL processen. Immers zoals in het vorige artikel is gesteld zijn de kosten van beheer en exploitatie ruwweg 66% van kosten van de IT. In het ITIL boekje Helpdesk [ITIL 1995] is aangegeven dat ruim 17% van de incidenten te wijten zijn aan de fouten in applicaties. Als de relatie tussen kwaliteitsattributen en de te hanteren SAM in kaart is gebracht, kan dit wellicht helpen om beheer en exploitatie goedkoper te maken.

> *Om de kosten van beheer en exploitatie te kunnen drukken moet dus iets aan de kwaliteit van de software worden gedaan.*

Dit kan bijvoorbeeld door te kijken naar een ingevulde SAM. Per onderkende relatie in de SAM moet dan worden gekeken welke kwaliteitsattributen van het Extended ISO 9126 model van toepassing zijn.

Zoals in het vorige artikel is gesteld zijn alle beheer en exploitatieprocessen te relateren aan de softwarelifecycle fasen. Ter vereenvoudiging zijn in dit artikel slechts 3 software lifecycle fasen gekozen te weten: Functioneel Ontwerp, Technisch Ontwerp en Realisatie. Voor deze 3 fasen is een vereenvoudigde SAM opgesteld en is gekeken naar een aantal in het oog springende SAM relaties die met de kwaliteitsattributen uit het extended ISO 9126 model zijn in te vullen. Figuur 2 geeft de gekozen SAM weer.

Lifecycle Stage >	**Functioneel Ontwerp**	**Technisch Ontwerp**	**Realisatie**
Service Function			
Service Delivery Set			
Service Level Management	X		X
Capacity Management		X	X
Contingency Planning			X
Availability Management		X	X
Cost Management			
Service Support Set			
Configuration Management			
Problem Management		X	X
Change Management	X		
Help Desk	X		
Software Control & Distribution			

Figuur 2, Vereenvoudigde SAM.

Voor de onderkende relaties tussen de ISO 9126 hoofdeigenschappen en de ITIL processen is gekeken welke fase van de lifecyle van toepassen zijn. Dit is opgenomen in Figuur 3.

ISO 9126 Hoofdeigenschap	**Belangrijkste ITIL processen**	**Belangrijkste LifeCycleFase**	**Argumentatie**
Functionele kwaliteit: - Bruikbaarheid - Functionaliteit	- Helpdesk - Change Management - Service Level Management	Functioneel Ontwerp	Kosten zijn gelegen in: - het aantal helpdesk calls - het aantal RFC's - de applicatiebeheerkosten. De functionaliteit moet overeenkomen met de business requirements zoals de Service Level Manager die heeft afgesproken met de gebruiker.
Technische kwaliteit: - Onderhoudbaarheid - Portabiliteit	- Problem Management - Availability Management - Contingency Management	Technische Ontwerp	Kosten in uitzoeken van de oorzaak van de fout kunnen verlaagd worden. Tevens zijn kosten van het niet kunnen gebruiken van een applicatie van belang. Het snel kunnen

			aanbrengen van een bug-fix kan de uptime verbeteren.
Infrastructuur kwaliteit - Betrouwbaarheid	- Problem Management - Availability Management - Contingency Management	Realisatie	Vermindering van verstoringen van de IT dienstverlening en snellere recovery van systemen leidt tot een hogere beschikbaarheid
Exploitatie kwaliteit - Efficientie	- Service Level Management - Capacity Management - Availability Management	Realisatie	Verbruik van resources kan geoptimaliseerd worden waardoor ook de beheerinspanning kan worden verminderd

Figuur 3, Relatie tussen ITIl, ISO 9126 en de softwarelifecycle

De relatering van ISO 9126 en ITIL en de vertaling naar de SAM mag alleen gezien worden als voorbeelduitwerking. Per bedrijf en project moet worden gekeken naar een eigen invulling.

De verschillende partijen (beheer, exploitatie, softwareontwikkeling en applicatiebeheer) moeten dus afspraken maken over welke kwaliteitsattributen worden gebruikt en erkend tijdens het traject. Als voorbeeld kan gekeken worden naar de focus van de technische kwaliteit. Om de kosten in exploitatie te drukken moet gedurende het technische ontwerp de technisch ontwerper regelmatig contact hebben met de Problem Manager om te kijken hoe het ontwerp zodanig kan worden opgezet, dat problemen snel te traceren en op te lossen zijn.

De SAM
De uitwerking uit de vorige paragraaf is opgenomen in Figuur 4.

Lifecycle Stage >	**Functioneel Ontwerp**	**Technisch Ontwerp**	**Realisatie**
Service Function			
Service Delivery Set			
Service Level Management	- Bruikbaarheid - Functionaliteit		- Betrouwbaarheid - Efficientie
Capacity Management		- Onderhoudbaarheid - Portabiliteit	- Betrouwbaarheid - Efficientie
Contingency Planning			- Betrouwbaarheid - Efficientie
Availability Management		- Onderhoudbaarheid - Portabiliteit	- Betrouwbaarheid - Efficientie
Cost Management			
Service Support Set			
Configuration Management			
Problem Management	- Bruikbaarheid - Functionaliteit	- Onderhoudbaarheid - Portabiliteit	- Betrouwbaarheid - Efficientie
Change Management			
Help Desk	- Bruikbaarheid - Functionaliteit		
Software Control & Distribution			

Figuur 4, Voorbeeld van een Service Activity Matrix (SAM), [ITIL 1993]

Per relatie in de SAM die is onderkend kan nu een checklist worden opgesteld waaraan de software moet voldoen en wel per kwaliteitsattribuutgroep. Dit kan uiteraard worden opgesplitst per kwaliteitsattribuut van de onderkende groepen.

Een voorbeeld uitwerking is gegeven voor het beheerproces 'Problem Management' in relatie met de fase Realisatie en wel voor de kwaliteitsattributengroep 'Onderhoudbaarheid'. Zoals in Figuur 1 is weergegeven, bestaat deze groep uit 6 kwaliteitsattributen die zijn opgenomen in Figuur 5. Voor elke ISO kwaliteitsattribuut is het belang aangegeven voor het Problem Management proces. Tevens is per attribuut aangegeven welke aspecten een rol spelen bij het opzetten van een checklist voor de betrokken applicatie. Een checklist is nodig om de kwaliteit van applicaties meetbaar te maken.

Een voorbeeld van een uitwerking van een aandachtspunt tot een checklist is bij voorbeeld het aandachtspunt ' Eenvoud van de programmeertaal' waarvoor een lijst kan worden opgesteld van bepaalde programmeertaalinstructies die niet gebruikt mogen worden (b.v. een GOTO statement). Ook kan hiermee gekeken worden of een bepaalde programmeertaal überhaupt wel acceptabel is voor het bedrijf in kwestie.

Kwaliteitsattribuut	**Belang voor Problem Management**	**Checklist aandachtspunten**
Analyseerbaarheid	- Snelheid van lokaliseren oorzaak	- Documentatie van keuze momenten die tijdens de realisatie zijn gedaan - Modulariteit van software - Hoeveelheid van annotatie bij de sourcecode - Informatie in foutboodschap zoals een uniek foutnummer en CI aanduiding
Wijzigbaarheid	- Snelheid van herstel van de dienstverlening	- Eenvoud van de programmeertaal - Voorkomen van exotisch applicatie constructies die (bijna) niemand nog kan uitpluizen - Gebruik bij client server applicaties stored procedures zodat aanpassen / tunen van SQL code wellicht (centraal) kan geschieden zonder hercompilatie van client applicaties. Distributie is dan ook erg eenvoudig.
Stabiliteit	- Voorkomen is beter dan genezen	- Robuustheid mag wel wat kosten in termen van responstijd. Liever 0.1 seconden extra checks per

		transactie dan veel incidenten. - Gebruik bij client server applicaties stored procedures zodat scheiding tussen client en server eenduidiger is - Controle op input van gebruikers - Integriteitsregels in het schema van de database opnemen in plaats van in client applicatie
Testbaarheid	- Applicaties moeten testbaar zijn om problemen op te sporen	De volgende zaken verhogen de testbaarheid: - regressietest - te testen in modulen - debugging mogelijkheid - foutregistratie, b.v. fouten wegschrijven naar een errorlog weggeschreven. Bij Windows NT b.v. naar de eventhandler
Beheerbaarheid	- Hoeveel resources kost een applicatie	- De code moet beschikbaar zijn - toegankelijk zijn - gedocumenteerd zijn
Herbruikbaarheid	- Hergebruik verkleint het volume van te analyseren software	- Schrijf componenten zodanig dat ze herbruikbaar zijn - Zorg voor een stabiele interface van componenten

Figuur 5 , Voorbeeld Kwaliteitsattributen uitwerking

De in Figuur 5 opgenomen aandachtpunten zijn algemeen gehouden. Wellicht dat in een bedrijf een algemene lijst wordt opgesteld en per ontwikkelomgeving een specifieke invulling wordt gemaakt.

Kosten / baten
Als bekend is welk beheer, exploitatie of applicatiebeheerproces het meeste kost, kan bij voorkeur dit proces het eerste worden uitgewerkt. In de meeste gevallen zal dit de helpdesk, problem management en applicatiebeheer betreffen. Natuurlijk kan voor bestaande applicaties alleen bij grote wijzigingen of herbouw iets structureels worden gedaan aan de verbetering van de kwaliteit. Het opzetten van de kwaliteitscriteria kost echter tijd en geld. Toch is het van belang dat er zo snel mogelijk aangevangen wordt met het opstellen hiervan.

De baten zijn te bepalen aan de hand van het verlies in geld dat slechte applicaties momenteel opleveren. Een goed ingericht helpdesk proces en availability proces moeten de verstoringen kunnen rapporteren waardoor het verlies te bepalen is.

Theorie en praktijk
De in dit artikel voorgestelde werkwijze om kwaliteit van software en serviceverlening te verbeteren en tevens de beheerkosten te verlagen zijn in de praktijk vaak moeizaam te realiseren. Hoe vaak krijgt een programmeur niet de volledige vrijheid binnen tijd en geld om de functionaliteit te bouwen? Veelal wordt door programmeurs gesteld dat hun creativiteit niet mag worden aangetast door regelgeving.

Het verdient dan ook aanbeveling deze checklists door de programmeurs zelf op te laten stellen. Hiervoor moet wel een kwaliteitsbesef worden gekweekt. Het is de moeite waard om te evalueren of een programmeur niet eerst eens 3 maanden mee moet lopen met een applicatie die in produktie is op zowel de helpdesk, problem management als applicatiebeheer. Zoiets als een kind met vuur confronteren voordat het zelf met vuur gaat experimenteren.

Probeer maar eens onder de druk van 1.000 gebruikers, die niet meer kunnen werken, een SQL statement van 1 pagina (zonder annotatie of inspringen) te doorgronden en te verbeteren zonder een nieuwe fout te introduceren. Dat een applicatie doet wat functioneel gevraagd is, is nog wel eens meer geluk dan wijsheid. Iedere applicatiebeheerder kent deze problemen. De programmeurs zouden dat ook eens moeten ervaren, ik denk dat dit ook significant kan bijdragen in het besparen van kosten in zowel de exploitatie- als de beheeromgeving.

Literatuur

[RADAR 1996] B. van Zeist e.a., “KWALITEIT VAN SOFTWAREPRODUKTEN”, 1e druk Kluwer Bedrijfswetenschappen, 1996, ISBN 90-267-240-6.

[ITIL 1993] CCTA, “SOFTWARE LIFECYCLE SUPPORT”, 2e dr.,Norwich, HMSO Publicity, 1989, ISBN 0 11 330559 1

[ITIL 1995] CCTA, “HELPDESK”, 5e dr.,Norwich, HMSO Publicity, 1995, ISBN 0 11 330522 2

[ROCCADE 1997] Beheer en vernieuwen van IT met R2C - Roccade Atribit , 1997, ISBN 90-803102-4-7