IVAN VERBORGH

IT BASICS

WAT EEN MANAGER HOORT TE WETEN

VRAGEN EN ANTWOORDEN VOOR DECISION MAKERS

LANNOO CAMPUS

D/2020/45/112 – ISBN 978 94 014 6714 8 – NUR 801, 980

Vormgeving omslag: Gert Degrande | De Witlofcompagnie
Vormgeving binnenwerk: Wendy De Haes

Uitgeverij LannooCampus maakt deel uit van Lannoo Uitgeverij,
de boeken- en multimediadivisie van Uitgeverij Lannoo nv.

Uitgeverij LannooCampus
Vaartkom 41 bus 01.02
3000 Leuven
België
www.lannoocampus.be

Postbus 23202
1100 DS Amsterdam
Nederland
www.lannoocampus.nl

INHOUD

ANDEREN OVER DIT BOEK

‘Toen ik Minister van Justitie werd, zag ik al snel dat de ICT-strategie nodig bijgestuurd moest worden. De digitalisering van justitie kon alleen maar succesvol zijn met een goed management. Het is op het ogenblik dat Ivan Verborgh de leiding kreeg over de stafdienst ICT, dat er belangrijke en zichtbare vooruitgang werd geboekt. Ik ben dan ook verheugd dat hij met dit boek zijn inzichten en aanpak wil delen met een ruimer publiek. De door hem uitgezette bakens voor de ontwikkeling van ICT in grote organisaties zullen nog lange tijd meegaan, en dat zowel binnen als buiten de overheid.’

— ***KOEN GEENS**, Vice-eersteminister, minister van Justitie en minister van Europese Zaken*

‘Dit boek reikt managers de basisbegrippen aan om in gesprek te gaan met IT-professionals, en dat is nodig. Want als een project pijnlijk de mist in gaat, is dat al te vaak te wijten aan een dovemansgesprek tussen ‘de business’ en ‘de IT’. Ivan Verborgh praat zowel de taal van de CEO als die van de CIO, en bezit het zeldzame talent om het ene in het andere te vertalen.’

— ***DOMINIQUE DECKMYN**, Technologiejournalist bij* De Standaard

‘Onze maatschappij verandert en schreeuwt nu meer dan ooit om een nieuw model. We leven in een digitaal tijdperk waarin iedereen en alles geconnecteerd is. Maar wat als ik nu geen technologisch expert ben? Dit boek geeft je een eenvoudig antwoord op alle vragen die je nooit hebt durven te stellen. Een manier om ook als niet-ingenieur de “taal van digitaal” te hanteren en op die manier je organisatie door transformatie te leiden.’

— ***SASKIA VAN UFFELEN**, Corporate VP Benelux GFI & Digital Champion Belgium*

‘IT is geen noodzakelijk hulpmiddel meer. Informatie- en communicatietechnologie is zo verweven met onze activiteit dat we zonder IT gewoon geen business meer hebben. Het is een wezenlijk onderdeel van al onze processen. Geen enkele eindverantwoordelijke van een organisatie kan het zich veroorloven om geen inzicht te hebben in de IT-basics. Dit boek geeft inzicht in de complexiteit van het IT-management in grote organisaties, kaart de relevante vragen aan die iedereen tegenkomt, geeft concrete antwoorden die op de praktijk geïnspireerd zijn en geeft tal van tips en tricks. Met als toetje de zeer handige overzichten van vragen en antwoorden. Een must voor elke manager.’

— ***HANS D’HONDT**, Voorzitter van de Federale Overheidsdienst Financiën*

‘Men zegt wel eens dat de gemiddelde leidinggevende een aandachtsspanne heeft die korter is dan die van een goudvis. Nou is dat natuurlijk (hopelijk) niet waar. Maar voor iedere topmanager met bar weinig tijd die wil weten wat de echt belangrijke overwegingen zijn rond ICT, is dit boek een absolute aanrader. En voor degenen die nog steeds nerveus worden van alles dat langer leest dan een tweet: de aparte pagina’s met Questions to Ask & Answers to Get brengen je pijlsnel naar de essentie.’

— ***RON TOLIDO**, Executive Vice President en Chief Innovation Officer, Capgemini*

‘De tijd van IT-experten en anderen die niets van IT kennen is voorbij. Het is onze verantwoordelijkheid om allen een basiskennis op te doen rond IT. Dit boek verlaagt de barrières om ermee te beginnen en geeft elke leider de kans om snel, en op een begrijpbare manier, de juiste taal en begrippen op te pikken. Zelf zal ik dit boek ook regelmatig weer opendoen om mijn kennis op te frissen.’

— ***PHILIPPE DE BACKER**, Minister van Digitale Agenda, Telecommunicatie en Post*

VOORWOORD

Voor wie is dit boek bestemd?

Dit boek is bestemd voor de CEO of leidinggevende die greep wil krijgen op de bijzonder technische wereld van de ICT (kort voor: informatie- en communicatietechnologie) en daarin wil kunnen meespreken op managementniveau.

Eerst worden de onvermijdelijke basics uiteengezet. Daarop bouwen we voort om inzicht te verschaffen in wat er wel en wat er niet toe doet, waar je het verschil kunt maken en welke ICT-gerelateerde aandachtspunten en opportuniteiten er horen te zijn in je organisatie.

Het boek biedt aanknopingspunten, direct in de materie en vanuit de ICT-praktijk aangereikt, zonder dat een volledige ICT-voorstudie nodig is. De technische uitleg wordt – zo veel mogelijk – beknopt en *high level* gehouden, zonder dat de samenhang of het goede begrip eronder gaan lijden. Waar nuttig wordt een historische context geschetst.

Het uitgangspunt van dit boek is ICT in een *bedrijfscontext*, en dus gaat het over goed functionerende programma's die gezamenlijk gebruikt worden. De onderdelen van een programma en hun onderlinge samenwerking worden uitvoerig besproken. Daarbij wordt ook de achterliggende complexiteit bij het maken van software toegelicht. De lezer krijgt handvatten aangereikt om de kwaliteit van een programma in te schatten, te borgen en fundamentele keuzes te kunnen maken tijdens een ontwikkelingstraject.

Vervolgens wordt het beschermen en beveiligen van de ICT-bedrijfsmiddelen toegelicht en aangevuld met een aantal IT-concepten en handige weetjes. Ook het reilen en zeilen in de IT-sector ontbreekt niet. Maar het boek heeft niet de pretentie om elk mogelijk (nieuw) onderwerp aan te pakken, wél om een goede basis te leveren in dit bijzonder boeiende domein.

Hoe is dit boek opgebouwd?

Elk topic wordt gekaderd in het grote geheel en zo beknopt mogelijk toegelicht, zonder dat er veel voorafgaande technische bagage noodzakelijk is.

Vervolgens formuleren we vanuit die toelichting een reeks relevante vragen (**The Question(s) to Ask © – TQ2A ©**) die leidinggevenden kunnen stellen om hierover een gesprek aan te gaan en de factfinding te starten of de stand van zaken op te maken.

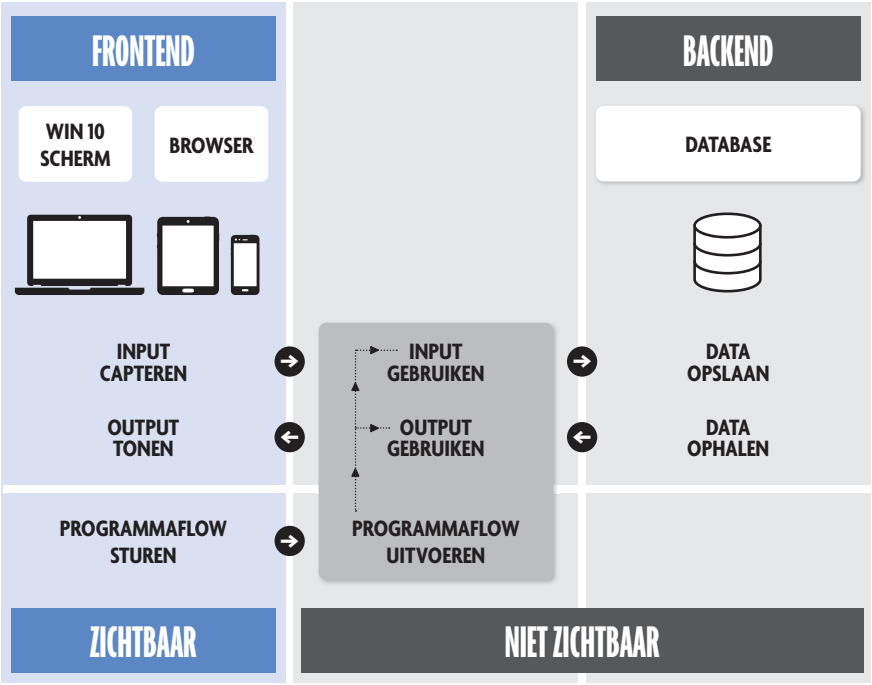
Tezelfdertijd worden er typeantwoorden (**The Answer(s) to Get © – TA2G ©**) bijgeleverd en wordt waar mogelijk ook een scala van antwoordmogelijkheden uitgewerkt, gaande van 'beperkt' tot 'volledig' antwoord. Daarmee kun je het verkregen antwoord toetsen en evalueren in het grotere geheel, en kunnen eventuele beslissingen worden genomen. Evengoed kan de conclusie zijn dat de situatie onder controle is en slechts opgevolgd moet worden, of dat er geen toepassingsgrond is.

De lezer kan dus meteen aan de slag met deze 'relevante vragen & mogelijke antwoorden', maar het is wel bijzonder aan te raden om eerst de topic **volledig** door te lezen, inclusief de eventuele doorverwijzingen, voor het geval er vragen ter verduidelijking komen.

LEESWIJZER: WAT JE *MOET* WETEN OVER PROGRAMMA'S (OF APPLICATIES) EN HUN ONDERDELEN

Dit stuk is van het verklarende type, waarin alle puzzelstukken worden geduid die verder in het boek worden uitgewerkt. Even doorbijten dus.

Intuïtief weet iedereen wat een softwaretoepassing of 'programma' is en kan doen. Voor een programma dat kleiner in omvang of functionaliteit is, wordt almaar meer de term **applicatie** of **app** gebruikt, in samenhang met het gebruik van nieuwe toestellen (**devices**) zoals smartphones en tablets, en niet enkel meer de klassieke pc. Microsoft Word is steeds een programma geweest, en nu ook een app. WhatsApp begon als app en wordt nu ook naar de pc getrokken – het is dan veeleer een **programma**. En zowat iedereen denkt bij SAP aan een (zwaar) enterprise resource planning-programma waarmee alle processen binnen het bedrijf worden ondersteund en niet direct aan een app. Die betekenisverschillen zouden ons te ver leiden, voor de rest van het betoog zal doorgaans de term 'programma' worden gebruikt.



Essentieel is dat een **programma** slechts één woord is, waarachter een **geheel** schuilt.

programma =
{ frontend + applicatielogica + database }

Frontend: het *zichtbare* gedeelte dat op pc, tablet of smartphone wordt getoond.
Bij wijze van voorbeeld: je persoonlijke medisch dossier. In de frontend zal een arts je gegevens (naam, voornaam, geboortedatum, geslacht enzovoort invullen ('write'), meestal vanop de pc op zijn kabinet. Een (afgeleide) frontend kan je gedeeltelijke inzage geven op een mobiele app ('read').

Applicatielogica: het (voor de gebruiker *niet zichtbare*) gedeelte achter de frontend dat het programmagedrag regelt in functie van een handeling of keuze die je maakt.

Wanneer je arts op de knop 'Rookgedrag' klikt, krijgt hij een ander scherm met invulmogelijkheden over je rookgedrag. [Sinds wanneer rook je? Hoeveel per dag? Enzovoort.]. Het gedrag van het programma (**program flow**) gaat dus een andere kant uit; zoniet staat het vakje 'niet-roker' aangevinkt op 'JA' en kunnen de gegevens bewaard worden. Applicatie- of applicatieve logica (**application logic**) kan dus heel zichtbare dingen doen, maar evengoed een moeilijke (achtergrond)berekening van een loopbaananciënniteit uitvoeren.

Database: het (eveneens *niet zichtbare*) gedeelte achter de applicatielogica dat ervoor zorgt dat alle gegevens netjes worden opgeslagen voor latere, nuttige exploitatie, rapportage of statistiek. In wat gemakshalve 'administratieve programmatuur of applicatie' heet en het leeuwendeel van de business-ICT-markt omvat, worden gegevens gestructureerd opgeslagen in een klassieke, 'relationele' database die bevraagd wordt door middel van **Structured Query Language (SQL)**.

In de tabel **PATIENT** in de database kan bijvoorbeeld staan:

			TYPE		SYNTAX	
Veld1	Naam	in	CHARACTERS	en	50 tekens lang	JANSSENS
Veld2	Voornaam		CHAR		35	JEAN-PAUL
Veld3	Geboortedatum		DATE		DD/MM/YYYY	31/04/2024

(Er bestaan nog andere types databases, daarover later meer.)

Om basisgegevens én rokersgedrag in jouw medisch dossier op te slaan, moet er dus eerst iets gebouwd worden in een frontend, die vervolgens vanuit de ingebouwde applicatieve logica een ander stuk programma, code of scherm oproept en waarmee je nieuwe gegevens kunt invullen (**insert**), aanpassen of aanvullen (**update**) of schrappen (**delete**). De structuur van de database (het **datamodel**) moet zo opgezet worden dat die gegevens vlot en met een minimum aan herhaling (*redundantie*) worden opgeslagen. En uiteraard moet jouw rookgedrag niet aan dat van je buurman gelinkt worden. De gegevens moeten volledig en onvervormd worden opgeslagen voor later gebruik.

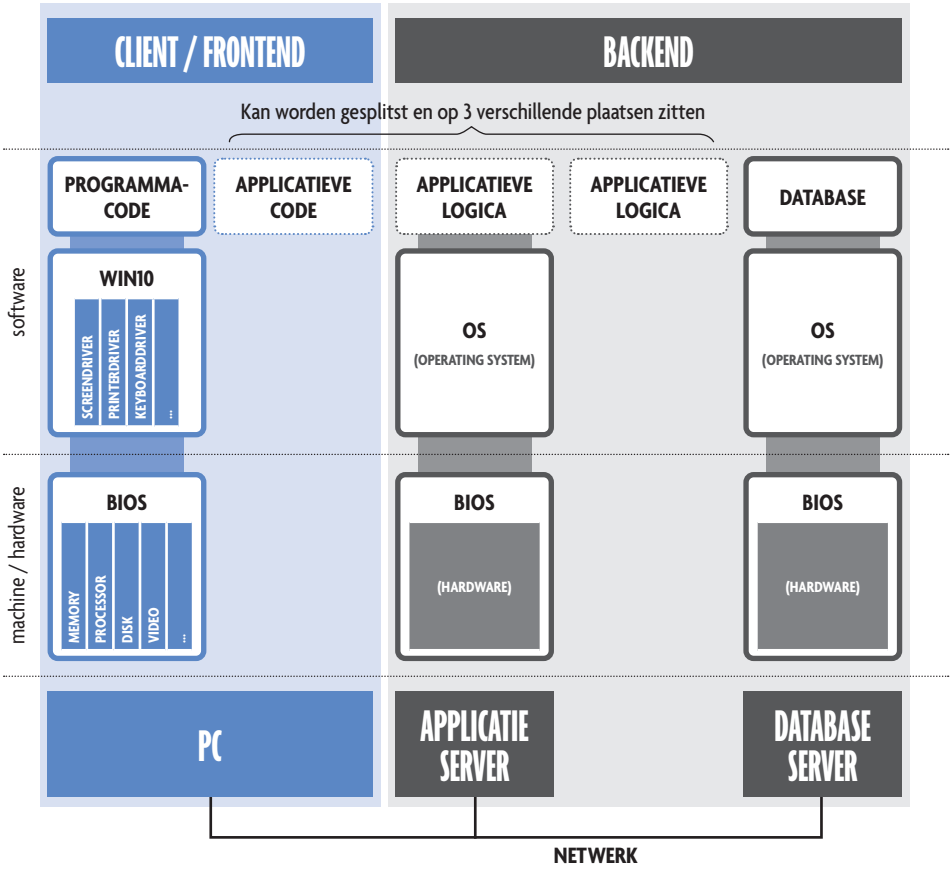
Al deze bouwstenen of 'building blocks' (frontend, applicatielogica en database) worden verder in dit boek dieper uitgewerkt, met heel wat bijbehorende TQ2A ©).

Terug naar het schema: één cruciaal element wordt meestal onderbelicht of gemakshalve vergeten omdat het triviaal is.

Alle building blocks staan nu apart getekend, maar moeten onderling wél met elkaar kunnen 'communiceren' en bijgevolg verbonden zijn. En die verbinding (over een **netwerk**) moet wél feilloos en performant werken.

werkend programma =
{ frontend + applicatielogica + database
+ onderlinge verbinding }

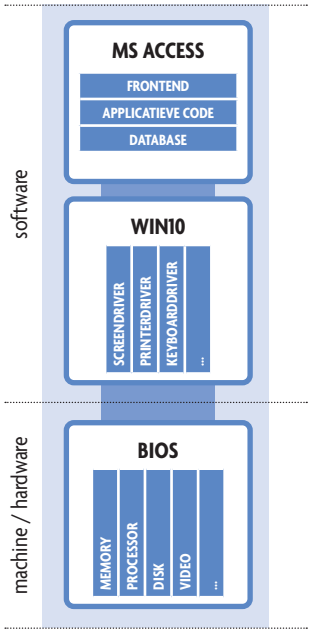
Zelfs dan nog zijn er heel wat variaties mogelijk in de onderliggende opbouw en samenhang.



In klassieke systemen met miljoenen gegevens (bijvoorbeeld bij de nationale belastingdienst) staat de frontend op een pc en de database op speciale, heel krachtige machines. Die worden onderling verbonden via een netwerk: het LAN (**local area network**), een WAN (wide area network) of nu ook het internet.

En waar staat de applicatie(ve) logica dan? Die kan (volledig of gespreid) op de frontend staan, op de database of ergens (deels of volledig) tussenin op een applicatieserver. Dat zijn keuzes die bij het ontwerp van het programma, al dan niet expliciet, gemaakt worden en die behoorlijk wat impact hebben op performantie en onderhoudsgemak van de programmacode.

In het schema zijn dit keer ook de noodzakelijke interne elementen getekend die een en ander doen functioneren. Omdat dit doorgaans allemaal stukjes zijn van *verschillende* leveranciers, wordt het *samen* doen functioneren van het geheel bijzonder complex. Daarbij komt dat ze elk apart dan eventueel ook nog fouten of **bugs** kunnen bevatten, of fouten *veroorzaken* in hun onderlinge samenhang. Wat meteen verklaart waarom het maken van 'goede' software niet echt eenvoudig is.



Mogelijk heb je zelf een eigen contactbeheerprogramma bijeengeklikt met MS Access. In dat geval is de frontend een schermpje dat onder Windows draait, staat de database in een Access-bestandsformaat op je *lokale* harde schijf in de pc, zit de verbinding intern in je machine tussen de softwarelagen en komt er *geen* externe netwerkverbinding aan te pas. De gegevens kunnen standaard ook niet door derden geraadpleegd worden. Zonder al te veel applicatieve logica, je zit tenslotte zelf aan het stuur en wat je zelf verzonnen hebt, voldoet altijd. Toch?

Willen we de contactgegevens binnen een groep kunnen gebruiken, dan moet de database *steeds bereikbaar* zijn en staat die beter *niet* op jouw lokale pc die uitgezet wordt aan het einde van de dag. Meer centraal op een machine die 24/24 beschikbaar is, en daarmee evolueren we richting het 'klassieke systeem'. Tegelijkertijd is er een mechanisme nodig om te voorkomen dat twee gebruikers die op hetzelfde contact gaan werken, bij het opslaan elkaars werk of input overschrijven. Zoiets moet geregeld worden, dat kan in de applicatiecode en ook via de database. Je gaat dan over van *single user* naar *multi-user*.

The Questions 2 Ask

- Welke programma's zijn cruciaal voor de organisatie?
- Volgens welk model zijn die opgebouwd?
- Quid levensduur, aantal releases, actuele ondersteuning ...?
- Zijn er (recurrente) problemen en van welke aard zijn die (technisch, contractueel ...)?

The Answers 2 Get

Minimaal:

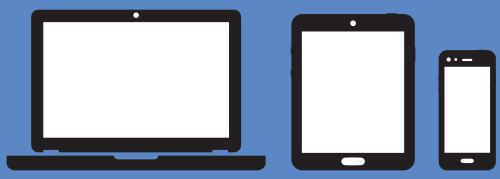
Een lijst van programma's.

Maximaal:

Een *volledige* lijst van programma's, met per programma:

- de gehanteerde aanpak of het achterliggende model,
- het aantal gebruikers,
- het volume aan data dat daarmee beheerd wordt,
- het belang / kritisch karakter voor de continuïteit van je business,
- een indicatie van de stabiliteit (zijn er regelmatig problemen mee?),
- de aard van de (contractuele) relatie met de leverancier,
- wanneer begon de ontwikkeling? Hoelang is het al in gebruik? In opeenvolgende versies?

én een goed zicht op de niet-officiële programma's die lokaal, onder de radar of door individuele gebruikers ontwikkeld werden (de zogenaamde 'shadow IT' – zie gelijknamige topic).



DEEL 1

PROGRAMMA (OF APPLICATIE): HET ZICHTBARE GEDEELTE (FRONTEND)

Eerst de 'look', dan de 'feel' en dan pas de rest

Een programma verschilt in zekere zin nauwelijks van een auto

Stel dat je een auto wilt kopen. De doordachte, analytische methode is een lijstje van basisvereisten op te stellen en daarmee fabrikanten en modellen nauwkeurig te gaan vergelijken. Maar het oog wil ook wel iets, best wel veel zelfs, en hoe die auto eruitziet is het eerste wat opvalt: de globale 'look' (en uiteraard de kleur). En vervolgens een proefrit om te voelen hoe die rijdt: de 'feel'. Als alles in balans is en het budget (de rest) is beschikbaar, dan ben je er. Zelfs als een auto aan elke mogelijke *rationele* eis voldoet en technisch perfect is, als je er niet mee gezien wilt worden, dan zul je hem ook niet kopen en dus niet gebruiken. Irrationeel, maar het is niet anders.

Dat geldt evenzeer voor een programma: de eerste kennismaking met de look-and-feel primeert, maar het is verre van eenvoudig om dat ook toe te geven. Al zeker niet door de erg rationele mensen die hard gewerkt hebben om alles goed te doen functioneren, vooral in de *onzichtbare* delen van applicatieve logica en database. En die dan afgerekend worden op het *zichtbare* gedeelte, op iets wat – in hun beleving – secundair en bijna irrationeel is en enkel te zien heeft met 'smaak' en niet met harde, solide techniek. Dit inzicht is cruciaal, want het gaat daarbij vooral over **user buy-in**.

Een programma dat rendeert, is een programma dat volop wordt gebruikt door de doelgroep waarvoor het ontworpen werd. En uiteraard correct doet wat het hoort te doen.

Daarbij is het basisscherm het eerste wat je ziet en tegelijk ook de voorbode van de (gebruikers)**interface, die de interactie tussen mens en computer mogelijk maakt en** die dwingend zal moeten gebruikt. Wringt de interface (te) veel, dan gaan de meeste mensen een link leggen met de kwaliteit van het *hele* programma. En dan gaat in de beleving heel het programma wringen, hoe goed de applicatieve logica of de achterliggende database ook in elkaar zit. Dit is een wijsheid waar onervaren programmeurs en designers helaas nogal eens tegen zondigen. Voor hen is die interface meestal wél oké, want ze hebben hem zelf verzonnen en dus geldt: 'Mijn kind, schoon kind.'

Meer dan dertig keer heb ik de 'The Rapid Application Development Race' ('The Original RAD Race ©') georganiseerd, een harde programmeerwedstrijd (tegenwoordig heet dat hackathon) waarbij teams van twee ervaren IT'ers, op basis van een onbekende, beschrijvende opgave en in volledig controleerbare omstandigheden, in twee dagen tijd een werkend programma moesten bouwen.

Een van de opvallendste lessen was dat – met *exact* dezelfde opgave en te bouwen functionaliteit – er visueel en gebruiksmatig heel uiteenlopende oplossingen werden neergezet.

Een programma met een **heldere interface** en flow, gemakshalve 'goede (programma)smoel' genoemd, maakte dat een nieuwe gebruiker na een halve minuut in staat was om zelfstandig aan de slag te gaan, zonder handleiding, zonder helpfunctie, zonder training. **Intuïtief** dus en nauwelijks of geen leercurve. En dat had relatief weinig van doen met de gebruikte programmeertaal, de ontwikkelmethodologie, de database en dergelijke, maar alles met het zich verplaatsen in de 'novice-gebruiker' en waarmee en hoe die logischerwijze aan de slag gaat. Als het programma dat goed voor elkaar heeft, dan is het pleit grotendeels gewonnen en is de buy-in bij de gebruiker verworven en krijgt het programma krediet bij de gebruiker, zelfs al mankeert er hier en daar wat. Zoniet is het programma (op termijn) gedoemd of voortdurend voorwerp van kritiek, hoe feilloos het achter de schermen ook mag werken.

Het 'betere' programma heeft trouwens een **novice-** en een **expertmodus** waartussen kan geschakeld worden en waarmee onderliggende opties *standaard* ingesteld staan dan wel volledig aanpasbaar worden. En het is flitsend snel.

The Questions 2 Ask

- Is er ervaring met programma's die een heel goede dan wel slechte acceptatie hadden? En die daardoor wel of niet gebruikt werden?
- Lag dat aan de gebruikersinterface op zich, dan wel aan het verschil tussen de (goede / slechte) interface met die van andere programma's?
- Of aan het gebrek aan ondersteuning (handleidingen, trainingen enzovoort)?

The Answers 2 Get

- Een lijst met programma's en een beschrijving van hun interface, met goede en slechte punten, wat zeker wel en wat geen navolging verdient.
- Dezelfde lijst, maar apart opgesteld en gescoord door de occasionele en de doorgewinterde gebruiker.
- Een definitie van de 'huisstijl', die perfect een verwijzing kan zijn naar de aanpak van een bepaalde bekende leverancier, bijvoorbeeld de Outlook-stijl.