Het Abstraction-Translation Paradigma en BI

Holistische blik op de werkelijkheid

Malcolm Chisholm

Enterprise Information Architecture blijkt een vage term, die tot grote frustraties leidt. In werkelijkheid is het een waardevol concept, maar uiterst complex. Een gebied dat speciale aandacht behoeft is Business Intelligence.

Er is een stroom aan bewijzen dat het ontbreken van oplossingen voor architectuur-problemen een negatief effect op BI heeft. Veel bedrijven hebben slechte ervaringen met BI. Ik doceer veelvuldig Master Data Management en daarbij zitten vaak projectmedewerkers voor datamarts en datawarehouses in de zaal. Hun belangstelling komt voort uit de problemen waar ze in BI-projecten tegen aanlopen, zoals roll-up rapporten die niet willen oprollen, maandelijkse rapporten die stabiel lijken maar af en toe merkwaardige, ongeloofwaardige data vertonen, en ad hoc query's die niemand snapt.

Deze BI-applicaties zijn ongetwijfeld technisch volledig in orde, maar wat ze produceren vertoont ernstige mankementen. Omdat deze gebreken voortkomen uit de data die in de datamarts zijn geladen, proberen de datamart-ontwikkelaars zich tegen de datastroom in een weg te banen, om de bron van de problemen te vinden en die vervolgens te repareren.
Dit is een uiterst moeilijke klus, zelfs als de locaties waar de problemen vandaan komen kunnen worden gevonden; meestal is de echte reden van de problemen niet boven water te krijgen. Bedrijven hebben nu eenmaal geen kaarten van hun data-landschap en de operationele systemen die de data produceren die door de BI-applicaties worden gebruikt, zijn te complex voor eenvoudige aanpassingen waarmee problemen kunnen worden opgelost die zich alleen voordoen in de BI-applicaties.

Architectuur: oplossing of probleem

Enterprise Information Architecture blijkt een vage term, die tot grote frustraties leidt. In werkelijkheid is het een waardevol concept, maar uiterst complex. Daardoor is het erg moeilijk vorm te geven of te visualiseren. Enterprise Information blijkt te bestaan uit vele verschillende dimensies of perpectieven, die op zich allemaal valide concepten zijn. Helaas correspondeert geen enkele daarvan met de werkelijkheid, en we zullen daar dus stuk voor stuk mee moeten afrekenen voor we tot een holistische blik kunnen komen.

Dat gesteld zijnd, is het mogelijk om de vraag te stellen of Enterprise Information Architecture de grote schuldige is voor het mislukken van BI-projecten. Het is waar dat het grootste stuk van de architectuur waar we mee te maken hebben is voor-bepaald. De verzameling legacy-systemen en aangekochte software-pakketten die de gemiddelde onderneming heeft draaien, lijken de omgeving te dicteren, die daardoor niet kan worden gepland. Dat is echter niet helemaal waar. Zelfs deze componenten zitten in een groter framework en de data die ze produceren kunnen in principe worden loskoppeld voor her-gebruik elders in de onderneming. Daarom zijn er probleem-gebieden binnen EIA die gemanaged kúnnen worden en die we eigenlijk zouden hóren te managen.

> Bedrijven hebben nu eenmaal geen kaarten van hun datalandschap

Afbeelding 1 toont zo'n probleemgebied, die ik het Abstraction-Translation Paradigma van EIA heb genoemd. Kort gezegd, visualiseert het de processen waarmee applicaties worden gecreëerd tijdens het passeren van opeenvolgende lagen van abstractie en vertaling, die er uiteindelijk voor zorgen dat de business informatie wordt opgeslagen als binaire representatie van feiten in geïmplementeerde databases. Vervolgens kunnen de data worden ingevoerd in datamarts voor gebruik in BI-applicaties. Het is echter nog steeds een binaire representatie van feiten. Om gebruikt te kunnen worden moeten die worden getransformeerd en vertaald door dezelfde lagen die nodig zijn voor het creëren van de operationele toepassing die het heeft voortgebracht.

Tenslotte verschijnt het opnieuw op business niveau als informa-tie. Gegeven de complexiteit van de wereldreis die de informatie aflegt, moeten we eigenlijk verbaasd zijn dat BI-applicaties überhaupt in staat zijn iets bruikbaars te produceren, in plaats van teleurgesteld te zijn in hun tekortkomingen.

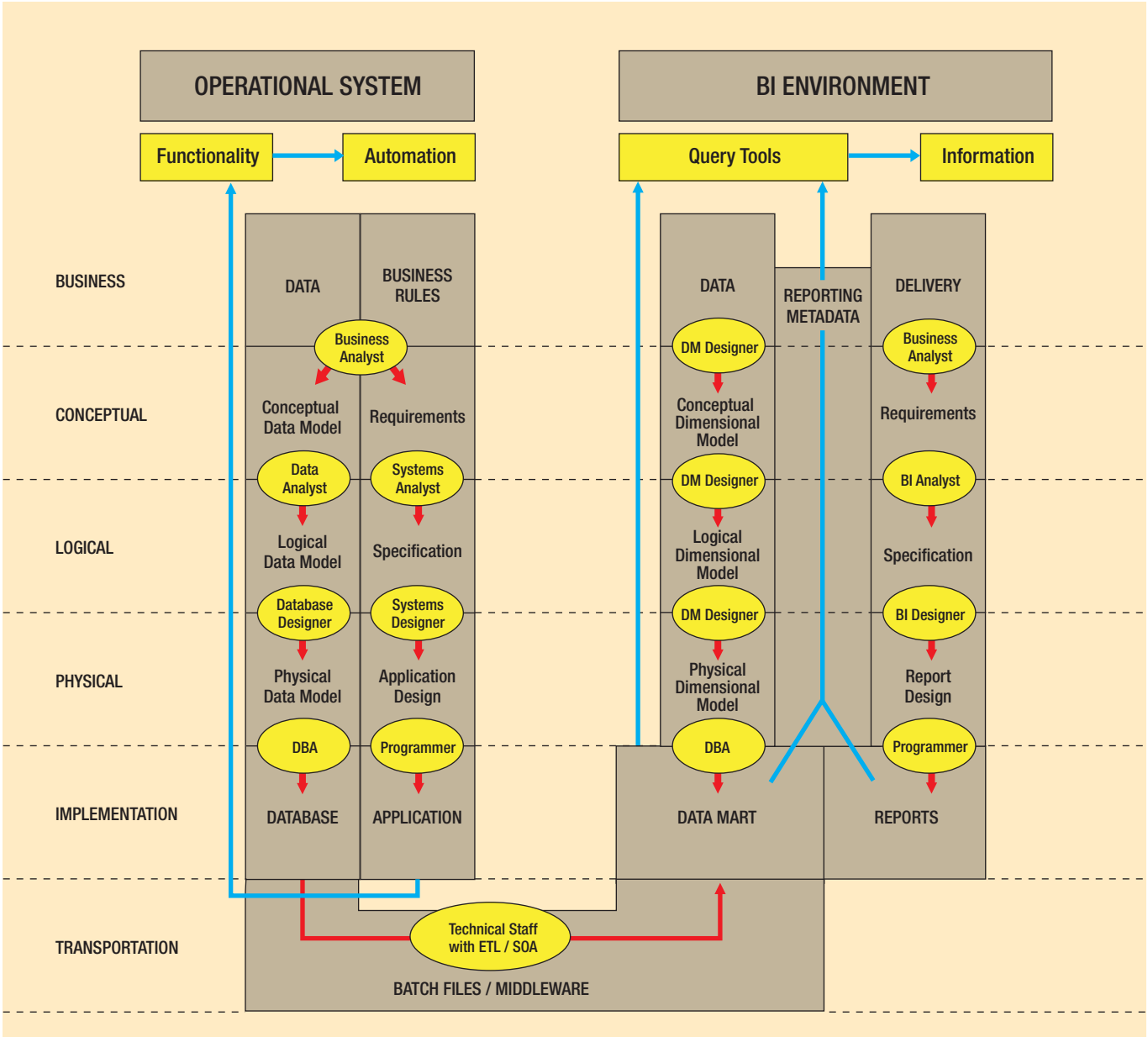
Het Abstraction-Translation Paradigma

De kolom aan de linkerbuitenkant in afbeelding 1 toont de analyse- en ontwerplagen die nodig zijn voor het bouwen van een operationele applicatie, en de rollen die gespeeld worden door de verantwoordelijken voor de gebeurtenissen in elke laag. Het is verdeeld in een kolom voor data en een voor business rules. Deze eindigen als respectievelijk fysiek geïmplementeerde databases en applicatie-logica.

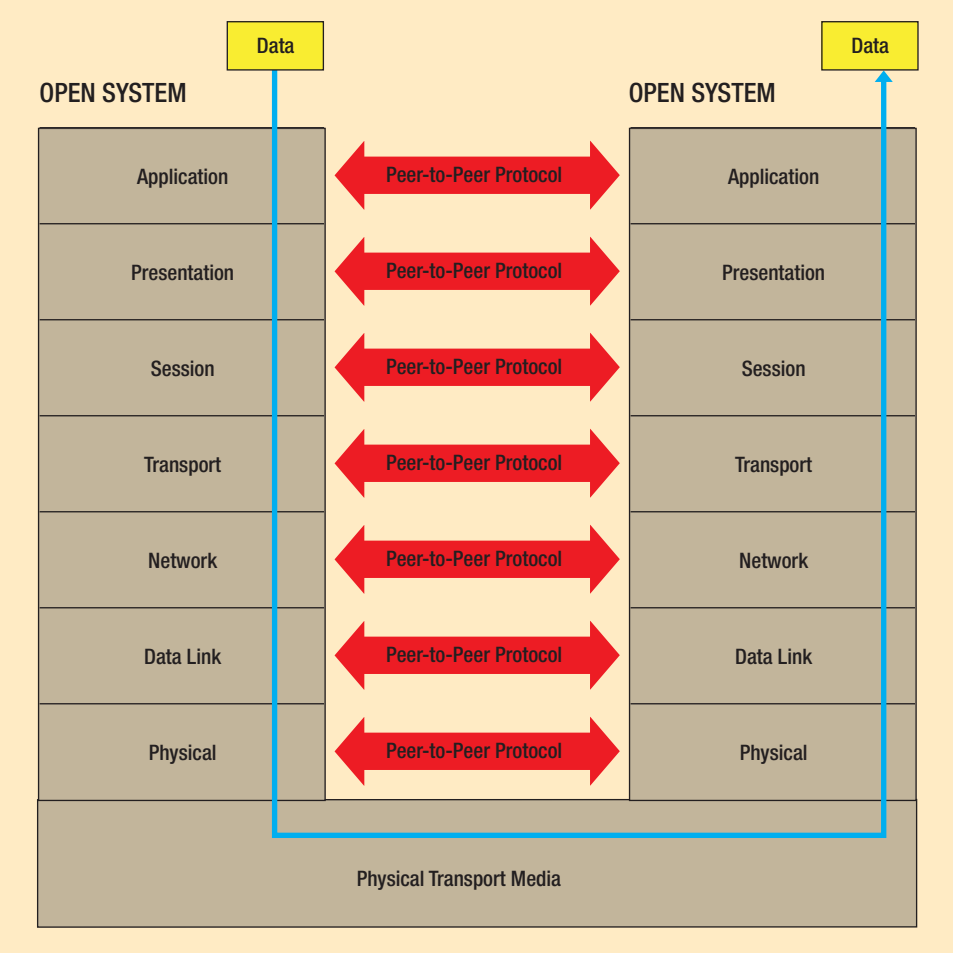
Het implementatieproces van een operationele applicatie begint in de business, waar een business analist vragen vanuit de business verzamelt, de huidige business processen beschrijft en de data identificeert die in deze processen worden gebruikt. Hieruit ontspringen idealiter use cases, workflows en een conceptueel datamodel met een glossarium van business begrippen. Deze representeren de business in wat in de afbeelding de 'Conceptual Layer' wordt genoemd. Helaas kent het begrip 'conceptual' vele definities, maar hier wordt er een directe representatie van de business mee bedoeld.

> In de datatransportlaag ligt de nadruk sterk op transport-concepten

Aan de datakant produceert een data-analist vervolgens een datamodel. Dat bevat een ontwerpelement. Zelfs als het om een actuele situatie gaat, zullen datamodelleerders altijd zeggen dat ze een plaatje leveren zoals de business de data echt 'ziet'.



Afbeelding 1: The Abstraction-Translation Paradigm (copyright Askget.com Inc).



Afbeelding 2: Het OSI Reference Model.

Wat het waarheidsgehalte van deze bewering ook zijn mag, we zien business concepten als vendors, klanten en medewerkers geabstraheerd worden naar 'party'-entiteiten, het bedenken van surrogaat-sleutels, merkwaardige namen van associatie-entiteiten, enzovoort. We hebben definitief het conceptuele niveau verlaten en betreden het logische niveau.

> Het probleem is dat de vastgehouden data opnieuw getransformeerd moeten worden door alle lagen heen

Aan de kant van de business rules haalt een systeem-analist datgene wat de business analist heeft voortgebracht tot op het kleinste detail uit elkaar en maakt een specificatie. Ook dit is niet puur analyse en bevat een ontwerp-element.

Dan komen we bij het fysieke niveau; dit is echt het domein van de ontwerpers. Aan de datakant is het doel een fysiek datamodel te maken, aan de kant van de business rules produceert men een applicatie-ontwerp. Als een commercieel software-pakket wordt aangeschaft, wordt meestal gedacht dat dit allemaal tevoren al gedaan is. Dergelijke software moet normaliter echter nog wel worden geconfigureerd om ze goed te laten werken in wat voor omgeving de klant ook heeft; dit is het equivalent van werken met een heel hoog niveau programmeertaal en een verzameling ontwerp-patronen. In de ideale situatie zijn het logische datamodel en de systeemspecificaties inputs in dit proces. Vertrekpunten zou een betere term zijn.

Dan worden de fysieke database en de applicatie gebouwd en overhandigd aan de DBA's en de productie-controle in de implementatie-laag. Deze technici kunnen de architectuur beïnvloeden door het nemen van beslissingen over de locaties van de database en de applicatie, de manier waarop query's fysiek worden uitgevoerd, enzovoort. Meestal hebben ze nauwelijks of geen begrip van de business omgeving die de door hun beheerde oplossing gebruikt. Hoe dan ook, de oplossing gaat aan het werk door de database met de applicatie combineren, met het verplaatsen van services en data door alle abstractie-lagen die gebruikt zijn om ze te ontwikkelen, en levert bruikbare functionaliteit aan de business gebruikers, meestal in de vorm van automatisering.

Uiteindelijk kunnen de data uit de applicatie naar andere plekken in de onderneming worden getransporteerd voor hergebruik. Datamarts zijn hiervan een mooi voorbeeld. In de datatransportlaag ligt de nadruk sterk op transportconcepten zoals XML en middleware. De betrokken technici geven weinig om data, ze voelen zich er niet verantwoordelijk voor. Als vrachtwagenchauffeurs die kisten moeten afleveren bekommeren ze zich om hun voertuig, de route naar de bestemming en hoe ze de kisten op hun bestemming moeten uitladen. Ze hebben nauwelijks idee over het belang van de inhoud van de kisten.

De implicaties voor BI

Dit is allemaal al vrij moeilijk, maar toch verwacht men tegenwoordig dat BI eenvoudig aan de architectuur kan worden toegevoegd. De rechterrij in afbeelding 1 toont een parallel proces voor de ontwikkeling van een BI-applicatie gebaseerd op een datamart. Hier zijn het de business vragen die het ontwikkelingsproces aandrijven. De techneuten zijn meestal specialisten in het werkgebied van datamarts, datawarehousing, BI-tools, enzovoort.
Rechts in de onderste laag bevindt zich echter de motor: IT-medewerkers die krachtige ETL-tools hebben of SOA/middleware, tools waarmee ze fysieke data kunnen pakken en neerzetten in een datamart. De technologie maakt het allemaal wel heel gemakkelijk. Verplaats de data naar een datamart, zet er een eindgebruikers query tool bovenop en klaar is kees. Vandaag de dag wordt alles natuurlijk steeds beter en het zijn straks de mensen die reporting metadatalagen implementeren die uitleggen wat de data betekenen, waar ze vandaan komen, enzovoort. Deze architectuurcomponent is nog zeldzaam, en werkt als het voorkomt maar gedeeltelijk, maar ik heb het toch opgenomen in de afbeelding.

Er zijn grotere problemen. Bij BI-oplossingen worden de data meestal ondergebracht in sterschema's die correleren met de belangrijkste query's die de behoeften van de business gebruikers definiëren. Dat dicteert hetzelfde ontwikkelingsniveau als van het operationele systeem, de conceptuele, logische, fysieke en tenslotte geïmplementeerde lagen passerend. Zo komt het ontwerp uiteindelijk de data tegen die getransporteerd worden vanuit de producerende operationele systemen.
Het probleem is dat in de BI-applicatie de vastgehouden data opnieuw getransformeerd moeten worden door alle lagen heen, tot het zinvol is voor de business, net als in het operationele systeem moest gebeuren. Er zijn twee moeilijkheden:

a. De BI-omgeving is meestal zo geconstrueerd dat het is gebaseerd op kennis van de informatiebehoefte van de gebruikers, maar met weinig of geen kennis van de transformaties die hebben plaatsgevonden in de ontwikkeling van de operationele oplossing;
b. De data in de database van het operationele systeem leiden een stuk van hun semantische eigenschappen af uit de transformaties die in bovenliggende lagen hebben plaatsgevonden, en een ander stuk uit de beperkingen die inherent zijn aan de applicatielogica, die langs een parallel pad heeft gereisd.

Deze eigenschappen zijn zo dus niet inherent aan de data zelf en kunnen er niet tegelijk mee worden vervoerd.
Geen wonder dat een kolom genaamd GROSS_SALES of een tabel genaamd CUSTOMER niet precies hoeven te betekenen wat de datamart-ontwerper denkt dat ze betekenen.

Abstraction and translation

Elk niveau in afbeelding 1 stelt een ander abstractieniveau voor. De term 'abstraction' betekent in de filosofie meestal dat ideeën gescheiden worden van objecten. In de afbeelding betekent het dat de concepten uit het vorige niveau worden gehaald die mappen naar de componenten die in het huidige niveau moeten worden gemanipuleerd. Bijvoorbeeld, een data-analist neemt de concepten uit het conceptuele datamodel – dat een tekstdocument kan zijn – en zet die in een datamodel in een CASE-tool. Elke laag in afbeelding 1 heeft zijn eigen verzameling componenten en concepten. Zo komt het dat de personen die binnen elke laag werken de neiging hebben een andere taal te spreken dan de mensen die in een andere laag functioneren.

> Het Zachman Framework spreekt aan omdat het lijkt aan te sluiten bij de werkelijkheid

Het abstraction-proces – het mappen van concepten in de vorige laag naar componenten in de huidige laag – wordt aangepast door het translation-proces en het jargon waarmee ideeën worden weergegeven in de vorige laag wordt vertaald naar het jargon van de huidige laag. Zo wordt de business 'informatie' het conceptuele 'data-element', dan het logische 'attribuut', dan de fysieke 'kolom' en ten slotte de 'data value' in de database, of misschien een XML 'document'. Deze keten illustreert hoe hetzelfde ding gemapped wordt naar verschillende concepten die beschreven worden in verschillende technische talen.
Groot probleem is hierbij dat de personen die in elke laag werken zich sterk focussen op de eigenaardigheden van de componenten waar ze mee te maken hebben en de tools die ze daarbij helpen. Een modelleerder van logische data zal zich zorgen maken over zaken zoals 'optionality' en 'cardinality', over welke notatie om ze in vast te leggen, en over de beslissing welke CASE-tool te gebruiken. Zulke zaken kunnen bijvoorbeeld een DBA die een database probeert te implementeren weinig schelen. Het is belangrijk dat de lagen elkaar een beetje overlappen, maar dat ze voor het grootste deel onderscheidend en uniek zijn.
De realisatie hiervan versterkt echter meestal de trouw die een technisch specialist heeft aan zijn eigen technische werkveld. Dit

werkt het behoud van het begrip van business data en business rules tegen.

Een vergelijking met het OSI Reference Model

Het Abstraction-Translation Paradigma zoals getoond in afbeelding 1 lijkt veel op het OSI (Open Systems Interconnection) referentiemodel voor de informatie-uitwisseling tussen open systemen. Dit wordt geïllustreerd in afbeelding 2 en verder uitgewerkt in ISO/IEC Standard 7498-1[1]. Het OSI-model beschrijft hoe de data die moeten worden gecommuniceerd door een aantal onderscheidende lagen gaan, die elk worden bestuurd door eigen standaards. Elke laag is bedoeld om met een verschillend niveau van abstractie van de afhandeling van de data om te gaan, zoals morfologie, syntax en semantiek.

> Het formele werk op de verschillende abstractieniveaus is helaas onontkoombaar

Uiteindelijk komen de data terecht in fysieke media waar het transport naar het ontvangende systeem plaatsvindt.
Messages die de data bevatten worden zo doorgegeven van de ene instance van een open systeem naar de ander, dezelfde lagen van abstractie doorlopend, totdat ze weer op het beginniveau zijn aanbeland (de applicatie). Hier kunnen ze worden behandeld in de termen van hun semantische inhoud. Dit model bevat ook peer-to-peer protocols tussen dezelfde niveaus in verschillende systemen. Er is geen equivalent van peer-to-peer protocols in het Abstraction-Translation Paradigma, omdat operationele systemen decennia eerder ontworpen en gebouwd kunnen zijn dan de BI-applicaties die nu hun data gebruiken. Het is onwaarschijnlijk dat de mensen die het operationele systeem oorspronkelijk hebben ontworpen nog onder ons zijn op het moment dat de BI-applicatie wordt geïmplementeerd en elke vorm van betrouwbare documentatie ontbreekt gegarandeerd.

Vergelijking met het Zachman Framework

Een ander concept dat overeenkomsten vertoont met het Abstraction-Translation Paradigma, is het Zachman Framework. Deze alom bekende matrix toont een ander aantal conceptuele niveaus gecorreleerd met data, functie, netwerk, mensen, tijd en motivatie. Het is bedoeld als framework voor de planning van activiteiten op het gebied van enterprise information architectuur.
Graeme Simsion heeft hierbij aangetekend dat er nauwelijks bewijsmateriaal voorhanden is, waaruit het succes van dit framework in de afgelopen 20 jaar blijkt[2]. Simsion geeft toe dat het framework hier en daar geadopteerd is, maar dat niet bewezen is dat het tot betere resultaten heeft geleid.

Toch spreekt het Zachman Framework aan omdat het lijkt aan te sluiten bij de werkelijkheid. Als de werkelijkheid die het framework beschrijft eigenlijk hetzelfde probleemgebied is als dat van het Abstraction-Translation Paradigma, is de kritiek van Simsion begrijpelijk. In plaats van proberen te werken met het Zachman Framework, en zo de problemen rond abstraction en translation te bestendigen, moeten we doen wat we kunnen om de verschillende lagen in het Abstraction-Translation Paradigma te ontdoen van tussenlagen.
Helaas is het doen van het formele werk op de verschillende abstractieniveaus onontkoombaar. De lagen in het Abstraction-Translation Paradigma en het Zachman Framework laten zien wat we moeten doen om informatiesystemen te bouwen. Als we abstracting en translation weglaten in de ene laag, moet het werk in een andere laag alsnog gedaan worden. In dat soort gevallen wordt het werk meestal maar half en dus niet goed gedaan, waar de kwaliteit van het product als geheel onder lijdt.

Het zal waarschijnlijk nog wel even duren voordat we een manier gevonden hebben om te kunnen ontsnappen aan het Abstraction-Translation Paradigma, zodat we rechtstreeks vanuit de business naar een implementatie kunnen gaan die zorgt voor herbruikbare data voor de rest van het bedrijf. Tot dat moment zullen we de dingen binnen het paradigma zo goed mogelijk moeten doen, op basis van de wetenschap dat het meer een aantal problemen definieert dan dat het een manier is om onze inspanningen beter te focussen. Het OSI-model suggereert wel dat we het beter kunnen doen op het gebied van peer-to-peer protocols binnen dezelfde laag. Wat ook een verbetering zou zijn, is als de abstraction- en translation-processen, met de bijbehorende artifacts, worden opgesloten in een te bouwen operationeel systeem, dat – misschien jaren later – beschikbaar is voor een datamart-bouwer. Maar we zullen eerst al deze uitdagingen onder ogen moeten zien, anders zijn we voor altijd gevangen in een zichzelf beperkende wereld van informatie management.

Dit is een bewerkte en vertaalde versie van de originele Engelse tekst, die u kunt vinden op onze website www.dbm.nl onder 'specials', 'extra materiaal'. In geval van discussies is de originele Engelse tekst doorslaggevend.

Noot

1. *http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip*
2. *http://tdan.com/view-articles/5279*

Malcolm Chisholm is directeur van Askget.com Inc te New Jersey.