Aanpak biedt goede ondersteuning voor testproces

Modelgedreven ontwerp van ETL-functies (3)

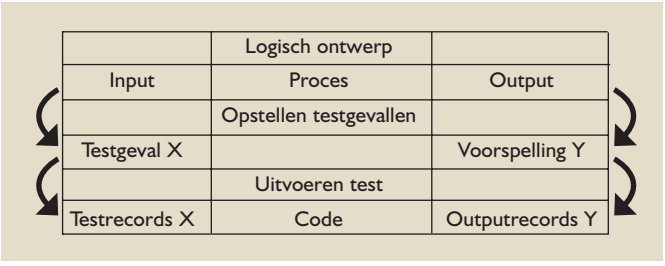
Mark Zwijsen en Rob Eveleens

In DB/M 6 van 2005 is de methode voor modelgedreven ETL-ontwerp uitgewerkt met een aanpak voor de vertaling van het functioneel ontwerp naar een technisch ontwerp of technische implementatie. In dit artikel wordt ingegaan op de bruikbaarheid van deze ontwerpmethode voor het testen van ETL-programma's.

Het testen van software is een essentieel onderdeel van het systeemontwikkelingsproces. Het kan kortweg worden omschreven als het toetsen van het geleverde product tegen de specificaties voor dit product. Bij het bouwen en testen van ETL-programma's is dit niet anders.
Het logische ontwerp van een ETL-programma is het startpunt voor zowel het bouwen van de ETL-mapping als voor het testen van deze mapping. De structuur van het logische ontwerp dient hiermee ook geschikt te zijn voor het ondersteunen van het testproces.

Beheersen

In het eerste artikel van deze serie is een overzicht gegeven van de kwaliteitsaspecten van een ontwerp. Onder het hoofdaspect 'Onderhoudbaarheid' wordt het aspect 'Testbaarheid' omschreven als "Het moet mogelijk zijn om een volledige set testgevallen uit het ontwerp af te leiden, die zo klein als mogelijk is en tegelijkertijd alle functionaliteit dekt", zie DB/M nr. 6 van 2004.
Om het testproces goed te kunnen beheersen, zijn dus compacte verzamelingen testgevallen en testgegevens nodig, waarmee de functionaliteit van het gebouwde ETL-programma afdoende kan worden getest.



Afbeelding 1: Samenhang tussen ontwerpen en testen.

In datawarehouse-omgevingen is het in principe geen optie om brongegevens sec te gebruiken voor het testen van de ETL-programma's. De redenen hiervoor zijn onder andere:

1. Testen betekent ook het voorspellen van het resultaat en het toetsen van deze voorspellingen. Dit voorspellen gebeurt 'op papier' tijdens de testvoorbereiding. Bij grote hoeveelheden gegevens is het onmogelijk om voor alle gegevens de resultaten te voorspellen.
2. Er is meestal sprake van zeer grote hoeveelheden gegevens. Het accent ligt dan eerder op een *stress of performance* test dan een test van de functionaliteit;
3. De bestaande bron bevat (nog) niet alle gevallen die ooit in de praktijk zullen voorkomen. Een aantal functionele onderdelen van het ETL-programma wordt hierdoor niet getest;

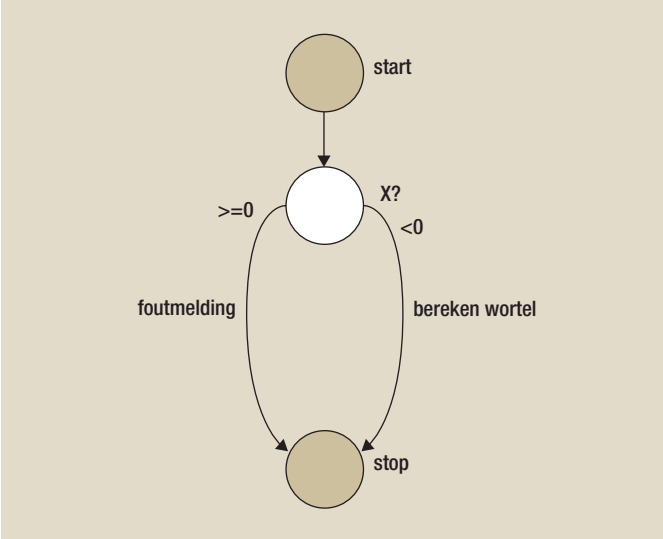
Er is dus een beperkte maar volledige set testgevallen nodig. Het vervaardigen van deze testset wordt in dit artikel uitgewerkt. Als opstapje maken we een vergelijking met een wiskundige functie.

Wiskundige functie

Een ETL-programma kan – gegeneraliseerd – worden beschouwd als de implementatie van een wiskundige functie F, waarop een formule y = F(x) van toepassing is. Hierbij staat x voor de invoerwaarde en y voor de uitvoerwaarde van de functie. Als voorbeeld wordt de vierkantswortelfunctie gebruikt, die op een rekenmachine is geïmplementeerd. De werking van deze rekenmachinefunctie wordt getest door het laten berekenen van een reeks uitkomsten, en deze te vergelijken met voorspelde uitkomsten.
Bij het samenstellen van een adequate verzameling testgevallen (paren van invoerwaarden en voorspelde uitvoerwaarden) moet met name gelet worden op alle 'grenswaarden', dat zijn de waarden waarbij een verandering van gedrag in de functie optreedt. Deze verandering is vooraf bekend, want deze volgt uit de gevraagde functionaliteit.
Bij een vierkantswortelfunctie is er één grenswaarde, namelijk 0:

- indien X >= 0, dan levert Y een geldige waarde op;
- indien X < 0, dan levert Y geen geldige waarde op (imaginaire getallen worden hier gemakshalve buiten beschouwing gelaten).

De verzameling testgevallen zou daarom in ieder geval de volgende {X,Y}-paren moeten bevatten: {1,1}, {0,0}, {-1,error} (ten behoeve van het voorbeeld wordt voor de X-waarde alleen met gehele getallen gewerkt).

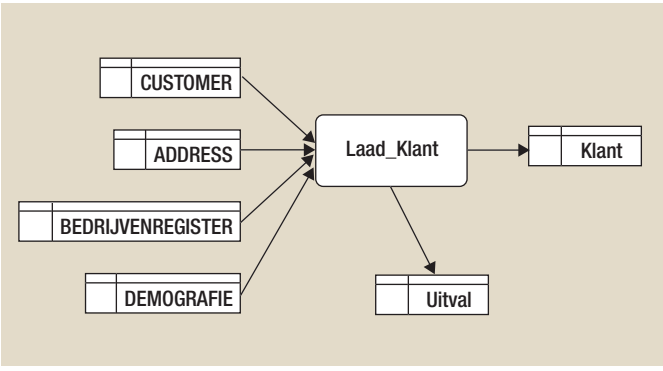


Afbeelding 2: Schema van vierkantswortelfunctie.

Verder is de omvang van de verzameling testgevallen redelijk arbitrair, want bij hoeveel testgevallen is voldoende aangetoond dat de rekenmachine zijn werk goed doet? Dit hangt geheel en al af van de gevraagde functionaliteit. Het al dan niet correct kunnen werken met gebroken getallen, heel kleine of juist heel grote getallen dient gespecificeerd te worden, en dan dienen er ook testgevallen te zijn die deze functionaliteit test.
Het gedrag van de functie kan worden weergegeven in de vorm van een schema, zie afbeelding 2. In dit schema bevindt één keuzemoment (het witte rondje), waar beoordeeld wordt of het linker- of rechterpad doorlopen moet worden. Dit hangt af van de waarde van X. In dit keuzemoment bevinden zich dus twee 'keuzeopties'.

Vertaling naar ETL

Als dit weer wordt terugvertaald naar een ETL-programma, dan representeert X de totale gegevensverzameling vóórdat het ETL-programma wordt uitgevoerd, en Y representeert de totale gegevensverzameling nádat het ETL-programma is uitgevoerd. Ook hier gaat het dus om een adequate verzameling testgevallen in de vorm van {X,Y}-paren, en dan met name om die gevallen die op en om de grenswaarden van X liggen.



Afbeelding 3: Laadproces Klant-dimensie.

Voor de verdere uitwerking wordt het voorbeeld gebruikt dat in het eerste artikel (zie DB/M 6, 2004) is gepresenteerd.
Een sterschema bevat een dimensie KLANT. Deze dimensie wordt opgebouwd uit tabellen CUSTOMER, ADDRESS, DEMOGRAFIE en BEDRIJVENREGISTER. Gegevens die niet door de validatie komen worden weggeschreven in de Uitvaltabel. Dit proces is schematisch weergegeven in afbeelding 3.

Het vaststellen van de verzameling testgevallen betekent nu het bepalen van de inhoud van de genoemde tabellen vóór en ná het uitvoeren van de functie Laad_Klant. In beide gevallen dient de inhoud van álle tabellen gedefinieerd te worden. De inhoud van de doeltabel is in veel gevallen namelijk ook input voor de functie. Dit is onder andere het geval indien de doeltabel een 'slowly changing dimension' bevat. Het gedrag van de ETL-functie zal anders zijn wanneer een bepaalde klant nieuw is in de doeltabel dan wanneer deze al bestaat. In het laatste geval is het gedrag afhankelijk van het al dan niet gewijzigd zijn van attributen van die klant.

Logisch transformatiemodel als basis

Biedt het modelgedreven ontwerp de juiste handvatten om de testgevallen op te stellen? We brengen het transformatiemodel (beschreven in het eerste artikel) nog even in herinnering en de tabel met transformatieregels die daar werd opgesteld. Een voor dit artikel bruikbare deelverzameling daarvan, beschrijft het verzamelen van de adresgegevens van belangrijke particuliere en zakelijke klanten en in het geval van zakelijke klanten ook hun Kamer van Koophandel nummer. Deze tabel is weergegeven in afbeelding 4. Hierbij zijn niet alle regels getoond, maar een deelverzameling die bruikbaar is voor het uitwerken van testgevallen. Ten opzichte van de beschrijving in het eerste artikel is één validatieregel (nr. 8) aan het transformatiemodel toegevoegd.

> Iedere verrijking wordt uitgevoerd binnen de scope van de trigger-entiteit

Hoewel de regels in deze tabel min of meer onafhankelijk zijn gespecificeerd, is er toch een zekere mate van afhankelijkheid tussen regels aanwezig, die invloed heeft op de wijze waarop deze regels omgezet worden naar testgevallen.
Met name in de verrijkingsregels is een afhankelijkheid waar te nemen, namelijk tussen regel 3 en 4. Immers, het doel van de verrijkingsregels is het verzamelen van extra attributen van de trigger-entiteit, in dit voorbeeld de CUSTOMER. Verrijking 4, welke een relatie beschrijft tussen ADDRESS en DEMOGRAFIE, zal uiteindelijk leiden tot demografieattributen van een CUSTOMER. Het al dan niet slagen van verrijking 4 wordt dus

mede bepaald door het al dan niet slagen van verrijking 3 met betrekking tot de adresgegevens. Het louter testen of een verrijking tussen de ADDRESS-tabel en de DEMOGRAFIE-tabel is gelukt, onafhankelijk van de relatie met CUSTOMER, kan tot foutieve conclusies leiden.
In het algemeen gesproken kan gesteld worden dat iedere verrijking wordt uitgevoerd binnen de scope van de trigger-entiteit, en dat het testen zich dus ook aan die scope dient te houden.

Uit deze scope-afhankelijkheid kan een rudimentaire volgorde worden afgeleid, waarin de verrijkingen logischerwijs worden uitgevoerd. Ook hiervoor geldt dat in de testgevallen deze volgorde terugkomt.
Verder is het voor het testproces van belang vast te stellen welke eisen gelden voor het rapporteren van mislukte validatieregels. Hierbij zijn in ieder geval de volgende twee alternatieven te onderscheiden:

1. De validatie vindt plaats in de volgorde van de validatieregels van het transformatiemodel, en alleen de eerste geconstateerde fout wordt gerapporteerd;
2. De validatie vindt altijd plaats op álle validatieregels, en de foutrapportage bevat informatie over álle validatiefouten.

Het moge duidelijk zijn dat het ene alternatief andere testgevallen met zich mee zal brengen dan het andere alternatief. Voor de uitwerking in dit voorbeeld gaan we uit van het eerste alternatief.

Stappen

De methode voor het vaststellen van de testgevallen kent de volgende stappen:

1. Zoek in de transformatieregels naar de keuzemomenten;
2. Bepaal per keuzemoment de mogelijke keuzeopties;
3. Maak een schema met de keuzemomenten en keuzeopties;
4. Bepaal de testgevallen door combinatie/aaneenschakeling van opties;
5. Creëer de testdata voor ieder van de testgevallen.

Bepalen keuzemomenten

Iedere transformatieregel kan een keuzemoment bevatten. In regel '5-Selecteren' bevindt zich een keuzemoment: de waarde van het CUSTOMER_NR. In regel '8-Valideren' bevindt zich ook een keuzemoment: het slagen van de verrijking met adres-gegevens. Over het algemeen bevat elke selectie- en validatie-regel een keuzemoment. In de andere regels kan ook een keuzemoment zitten, maar dit is niet noodzakelijk. Indien in één transformatieregel meer dan één keuzemoment gevonden wordt, kan dit erop duiden dat deze transformatieregel te complex is en moet worden opgesplitst.
Een extra keuzemoment is de afhandeling van historie. Verondersteld wordt dat de doeltabel een volledige Type 2-historie heeft. Het gedrag van de ETL-functie wordt hier beïnvloed door de inhoud van de eventueel reeds aanwezige versie van een klantrecord.

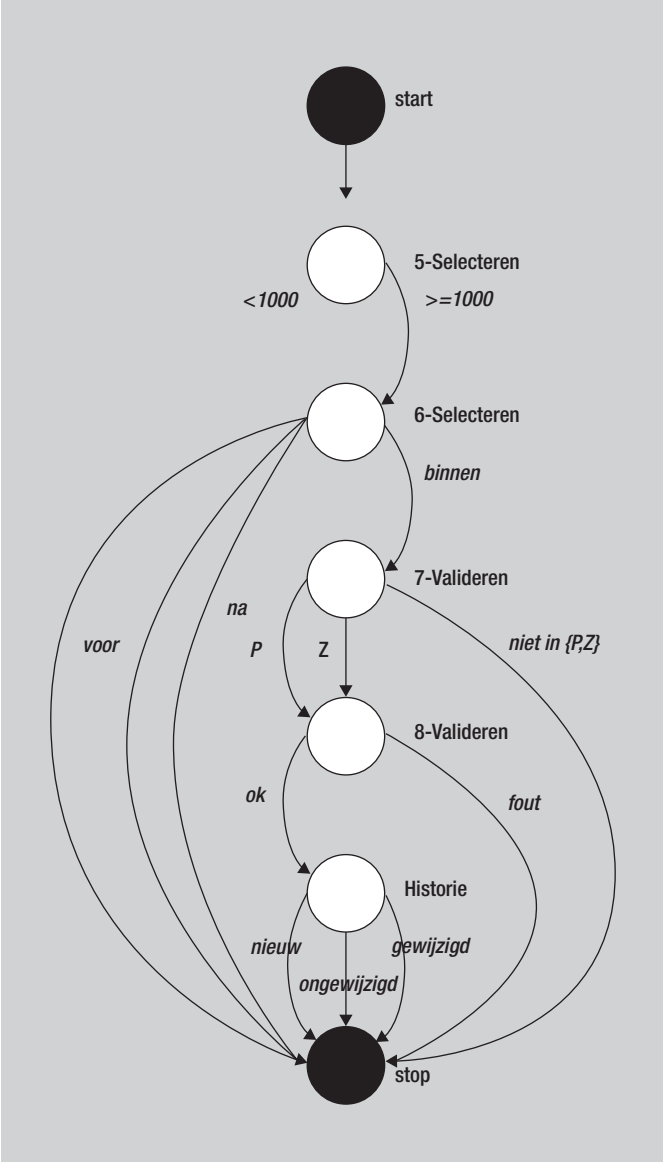
Bepalen keuzeopties

Bij ieder keuzemoment dienen te keuzeopties te worden bepaald. Vaak zijn dit er twee, omdat het dan om een ja/nee ofwel waar/onwaar situatie gaat. In de gevallen dat er op een waarde geselecteerd of gevalideerd wordt, kunnen er meer dan twee keuzeopties zijn. Bij het keuzemoment bij '7-Valideren' kunnen drie keuzeopties worden onderscheiden: {gelijk aan 'P'}, {gelijk aan 'Z'}, {Niet in ['P','Z']}. Maar ook dit is arbitrair, opties kunnen worden samengevoegd of gesplitst. De triviale aanname is: 1. dat verrijkingsregels nooit tot foutmeldingen of afbreken van het proces leiden: dat hoort pas bij validaties beschreven te staan; en 2. dat de volgorde van selecties niet relevant is.
Bij het keuzemoment 'Historie' worden drie keuzeopties onderscheiden:

1. De betreffende klant bestaat nog niet in de doeltabel;
2. De klant bestaat al wel en alle attribuutwaarden komen overeen;
3. De klant bestaat al en één of meer attribuutwaarden komen niet overeen.

Regel	Stap	Criterium	Evt. Actie
2	Verrijken	CUSTOMER - BEDRIJVENREGISTER: CUSTOMER.IDENTIFICATION = BEDRIJVENREGISTER.KVKNUMMER mits CUSTOMER.PRIVATE_BUSINESS_FLAG = 'Z'	Anders blanco
3	Verrijken	CUSTOMER - ADDRESS: CUSTOMER.ADDRESS_NR = ADDRESS.ADDRESS_NR -en- CUSTOMER.TIMESTAMP >= ADDRESS.VALID_FROM -en- kies MAX(ADDRESS.VALID_FROM)	
4	Verrijken	ADDRESS - DEMOGRAFIE: ADDRESS.POSTCODE = DEMOGRAFIE.POSTCODE	Anders blanco
5	Selecteren	CUSTOMER.CUSTOMER_NR >= 1000	
6	Selecteren	CUSTOMER.TIMESTAMP Binnen [VerwerkingsPeriode]	
7	Valideren	CUSTOMER.PRIVATE_ BUSINESS_FLAG = 'Z' of 'P'	Uitval
8	Valideren	Verrijking 3 is geslaagd	Uitval

Afbeelding 4: Transformatieregels Laad_Klant.



Afbeelding 5: Schema voor Laad_Klant.

Met behulp van deze keuzemomenten en keuzeopties kan nu een schema worden gemaakt. Het is belangrijk te vermelden dat dit schema niet hetzelfde is als een stroomschema. Een stroomschema visualiseert al een specifieke volgorde van bewerkingen en beslissingen. In de gebruikte ontwerpmethode wordt een vaste volgorde wordt aangehouden voor de verschillende transformatiestappen. Zo worden alle selecties pas beschreven na alle verrijkingen. In ons voorbeeld zijn de selectiecriteria (CUSTOMER_NR >= 1000 en TIMESTAMP in verwerkingsperiode) niet afhankelijk van de verrijkingsregels. In het uiteindelijke ETL-programma zullen deze selecties waarschijnlijk vooraan worden geprogrammeerd. De volgorde in het schema volgt de volgorde in het logisch transformatiemodel.

Het schema voor de ETL-functie is weergegeven in afbeelding 5. Juist als het aantal beslissingen groter wordt en de functionaliteit complexer wordt, is gebleken dat dit schema het goede begrip van de functionaliteit in samenspraak met bouwer, ontwerper en tester kan verzekeren.

Het schema in afbeelding 5 bevat op overzichtelijke wijze alle stappen uit het logische ontwerp. Het verder uitwerken van dit schema naar logische testgevallen en testrecords is een bekende werkwijze voor testers en wordt hierna verder uitgewerkt. Voor dit artikel is relevant hier te constateren dat de voorgestelde ontwerpwijze eenvoudig om te zetten blijkt naar een overzichtelijk schema, waaruit logische testgevallen zijn af te leiden.

Testgevallen

Een testgeval komt overeen met een route van 'start' naar 'stop' door het schema. Het aantal theoretisch verschillende routes in afbeelding 5 is 11. Dit is een relatief klein aantal. Te bedenken valt namelijk dat voor elke route testinput en testoutputvoorspelling moet worden gespecificeerd, en naarmate het aantal routes toeneemt wordt dit karwei omvangrijker. Nu is hier nog sprake van een zeer eenvoudige ETL-functie. De meeste ETL-functies zijn veel groter, en met het aantal keuzemomenten en keuzeopties groeit het aantal mogelijke routes zeer snel. Er zijn additionele criteria nodig om vast te stellen welke subset van testgevallen de functionaliteit voldoende afdekt. Enkele van deze criteria zijn:

1. Iedere keuzeoptie moet in ten minste één testgeval/route voorkomen;
2. Elke mogelijke combinatie van binnenkomende en uitgaande acties rond een keuzemoment moet in een logisch testgeval voorkomen
3. Niet iedere route zal in zijn geheel doorlopen worden, met andere woorden het 'bewandelen' van sommige keuzeopties zal direct leiden tot een eindsituatie (bijvoorbeeld het niet slagen van een verrijking of een selectie);
4. De 'hoofdroutes', dat zijn de routes waar het meeste verkeer langs zal komen, moeten het meest nauwkeurig worden getest.

Een ETL-programma werkt in veel gevallen batch-georiënteerd

Door de keuzeopties als in het dominospel te combineren kunnen de logische testgevallen worden geconstrueerd. Deze testgevallen zijn weergegeven in afbeelding 6. Bij ieder testgeval wordt een omschrijving gemaakt, plus een bondige omschrijving van het gewenste resultaat (de Y). Wanneer geen zinvolle omschrijving voor een testgeval gemaakt kan worden, kan dit wijzen op onduidelijke of tegenstrijdige logische specificaties.

Testdata aanmaken

Op basis van de testgevallen kunnen nu de testgegevens worden gemaakt, oftewel de paren {X,Y}. Eén X-waarde bestaat uit een

Testgeval	Omschrijving	Gewenst resultaat
Start / 5< / Stop	Niet relevante Customer	Wordt niet verwerkt
Start / 5>= / 6 voor/ Stop	Relevante Customer, wijzigingsdatum vóór verwerkingsperiode	Wordt niet verwerkt
Start / 5>= / 6 na / Stop	Relevante Customer, wijzigingsdatum na verwerkingsperiode	Wordt niet verwerkt
Start / 5>= / 6= / 7 niet(P,Z) / Stop	Deze customer heeft een onbekend type	In uitvalrapportage
Start / 5>= / 6= / 7P / 8+ / nieuw / Stop	Relevante Particuliere Customer, bestaat nog niet in doeltabel	Nieuw record in doeltabel
Start / 5>= / 6= / 7P / 8+ / ongewijzigd / Stop	Relevante Particuliere Customer, bestaat al in doeltabel met dezelfde gegevens	Geen wijzigingen in doeltabel
Start / 5>= / 6= / 7P / 8+ / gewijzigd / Stop	Relevante Particuliere Customer, bestaat al in doeltabel met afwijkende gegevens	Mutatie van bestaand record in doeltabel
Start / 5>= / 6= / 7Z / 8- / Stop	Relevante Zakelijke Customer, adres niet gevonden	In uitvalrapportage
Start / 5>= / 6= / 7Z / 8+ / nieuw / Stop	Relevante Zakelijke Customer, bestaat nog niet in doeltabel	Nieuw record in doeltabel
Start / 5>= / 6= / 7Z / 8+ / ongewijzigd / Stop	Relevante Zakelijke Customer, bestaat al in doeltabel met dezelfde gegevens	Geen wijzigingen in doeltabel
Start / 5>= / 6= / 7Z / 8+ / gewijzigd / Stop	Relevante Zakelijke Customer, bestaat al in doeltabel met afwijkende gegevens	Mutatie van bestaand record in doeltabel

Afbeelding 6: Tabel met testgevallen.

record in de trigger-tabel, eventuele bijbehorende records in de overige brontabellen en in de doeltabel. De trigger-tabel is de brontabel die de feitelijke kandidaten levert. De bijbehorende Y-waarde bestaat uit eventuele records in de doeltabel en/of de uitvaltabel.

Door het zorgvuldig kiezen van de attribuutwaarden van de records van de testgevallen, is het vaak mogelijk om alle testgevallen samen te brengen in één complete set testrecords als X-verzameling en één complete set records als Y-verzameling. Een ETL-programma werkt immers in veel gevallen batch-geöriënteerd. Na het uitvoeren van het ETL-programma is dan in één oogopslag duidelijk welke testgevallen goed en welke niet goed door de test komen.

Conclusies

Het modelgedreven ontwerp voor een ETL-programma biedt de juiste structuur voor het effectief voorbereiden en uitvoeren van tests. Door gebruikmaking van schema's is een dekkende set testgevallen samen te stellen. Zowel de gegevensverzameling vóór de testuitvoering (X) als die ná de testuitvoering (Y), kunnen op basis van deze testgevallen worden samengesteld. De ontwerpen voldoen hierdoor aan een belangrijk kwaliteitscriterium, de 'test-baarheid', welke weer onderdeel is van het meeromvattende criterium 'onderhoudbaarheid'.

Mark Zwijsen en **Rob Eveleens**
Mark Zwijsen (mark.zwijsen@atosorigin.com) en Rob Eveleens (rob.eveleens@atosorigin.com) zijn beiden Senior Consultants Datawarehousing bij Atos Origin.