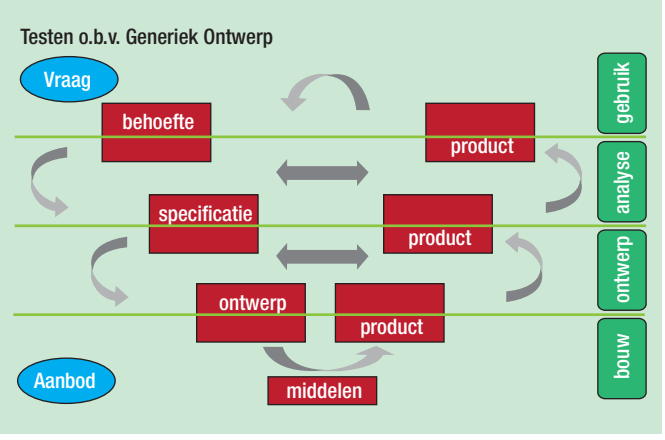
Gestructureerd afleiden van logische testgevallen

Testgraaf communiceert het ontwerp

Rob Eveleens, Kees Molenaar, Yvonne Schurink, Mark Zwijsen

Vrijwel iedere datawarehouse-implementatie bevat een ETL-component. Een ETL-programma onttrekt gegevens uit een bronsysteem (Extract), transformeert het in de gewenste vorm (Transform) en laadt de getransformeerde gegevens in een doelsysteem (Load).

Binnen de meeste datawarehouse-projecten wordt een zeer groot aantal ETL-programma's ontwikkeld. Een gemiddeld datawarehouse bevat al snel tientallen, zo niet honderden tabellen (feiten, dimensies, aggregaten), en het aantal ETL-programma's in een datawarehouse overstijgt vaak het aantal tabellen dat in dat datawarehouse aanwezig is. Dit komt omdat er vrijwel altijd voor gekozen wordt om per tabel een apart ETL-programma te bouwen dat het laadproces van die tabel verzorgt. En in veel gevallen blijft het niet bij één laadproces per tabel. Indien een tabel gegevens bevat die afkomstig zijn uit meer dan één bronsysteem, dan zal er per bronsysteem een apart laadproces worden gebouwd. Een datawarehouse is gedurende diens gehele levensduur aan verandering onderhevig. Door veranderende eisen van gebruikers, veranderingen in de bedrijfsprocessen of in de markt waarin het bedrijf opereert, zijn aanpassingen of uitbreidingen aan het datawarehouse nodig. Dit betekent dat er nieuwe tabellen en ETL-programma's gebouwd worden en dat er voortdurend bestaande programma's aangepast worden. Een efficiënte methodiek voor het ontwerp, bouw en het testen van ETL-programma's levert dus winst op voor een datawarehouse-project.



Afbeelding 1: Het ontwikkel- en implementatieproces.

Eerder publiceerden wij over het efficiënt ontwerpen van ETL-programma's, zie [1]. Nu zoomen we in op het efficiënt testen van ETL-programma's op basis van het detailontwerp. De testgraaf visualiseert een detailontwerp dermate sterk dat direct betrokkenen (ontwerper en bouwer) en indirect betrokkenen (opdrachtgever en beheerder) er baat bij hebben. De testgraaf zelf voegt daarmee waarde toe aan het project en de projectdocumentatie.

Logische versus technische testgevallen

Het ontwerp- en bouwproces van testgevallen is in wezen niet anders dan dat van 'andere' gegevensverwerkende systemen en ETL-processen:

- Het logische niveau richt zich op de structuur, de betekenis en de samenhang van de gegevens en gegevensbewerkingen, zonder enige wetenschap van, of rekenschap met de consequenties van de te gebruiken hulpmiddelen;
- Het technische niveau ontstaat uit een transformatie uit het logische model, die vervolgens implementatie-specifieke aanpassingen ondergaat, onder andere voor het optimaliseren van opslag en CPU en/of I/O performance of gebruiksgemak.

Voor testgevallen is het onderscheid dan:

- Logische testgevallen beschrijven dus te testen situaties in betekenisvolle termen;
- Technische testgevallen betreffen de verzameling van gewenste tabelwaarden als vertaling van de logische testgevallen.

Bij het testen van ETL-programma's wordt deze scheiding tussen een logisch niveau (en logische testgevallen) en een technisch niveau (en dus technische testgevallen) veel te weinig toegepast. De ontworpen testgevallen die we in de praktijk tegenkomen zijn vaak al verweven met de toegepaste architectuur en de gekozen ETL-tools. Dat heeft negatieve gevolgen voor de bruikbaarheid, onderhoudbaarheid en overdraagbaarheid. Een verzameling van logische testgevallen per ETL-proces is van groot belang, zowel voor projecten in verband met de decharge, als voor het beheer.

Het ontwerpen van logische testgevallen en het benutten van die inspanning binnen het project is een onderbelicht terrein waar veel winst valt te behalen. Het is sterk aan te bevelen de logische testgevallen en testgraaf met de bouwer en ontwerper af te stemmen en het detailontwerp op discussiepunten te verhelderen. Vaak blijken de vragen van de tester en bouwer weer te leiden

Bruikbaar	Volledig	De aanpak moet ervoor zorgen dat alles wat in het ontwerp wordt beschreven ook getest wordt.
	Nauwkeurig	De aanpak voorkomt interpretatieverschillen; alles wat nodig is om logische testgevallen te kunnen beschrijven dient beschreven te zijn.
	Consistent	De aanpak mag zichzelf niet tegenspreken.
	Efficiënt	De aanpak leidt naar een volledige en juist voldoende verzameling van logische testgevallen die de in het detailontwerp beschreven functionaliteit testen.
	Toegankelijk	De aanpak moet (1) aansluiten bij de kennis van de bouwer, de ontwerper, tester en beheerder en (2) op zichzelf gelezen en begrepen kunnen worden.
Onderhoudbaar	Testbaar	De constructie van de logische testgevallen is navolgbaar en eenduidig.
	Gestructureerd	De aanpak biedt een duidelijke structuur die ondersteunt bij het construeren van de logische testgevallen.
	Begrijpelijk	De gebruikte aanpak sluit aan bij de kennis van de tester.
	Wijzigbaar	Wijzigingen kunnen zonder inbreuk op de structuur en begrijpelijkheid worden aangebracht.
	Flexibel	Uitzonderingen kunnen zonder al te grote inbreuk op de structuur worden beschreven.
Overdraagbaar	Platformonafhankelijk	De verzameling van logische testgevallen kan zonder wijzigingen voor andere hulpmiddelen voor applicatie-ontwikkeling worden toegepast.

Afbeelding 2: Kwaliteitscriteria toegepast op methodieken om logische testgevallen af te leiden.

tot vragen aan de opdrachtgever over de beoogde functionaliteit. De testgraaf is daarbij een middel om toe te lichten welke functionaliteit nog nader gespecificeerd moet worden.
Het logische niveau voor testgevallen wordt opgebouwd uit het detailontwerp, waarbij het detailontwerp wordt vertaald naar:
1. Beslissingen (criteria en mogelijke uitkomsten);
2. Gegevensbewerkingen;
3. Een volgorde van bewerkingen en beslissingen.

In het ontwikkelproces wordt veel getest om te verifiëren of de producten overeenkomen met de behoefte, de specificatie of het ontwerp. De testgraaf wordt opgesteld op basis van het ontwerp, zie afbeelding 1. Op basis van de testgraaf wordt het product van de bouw getest.

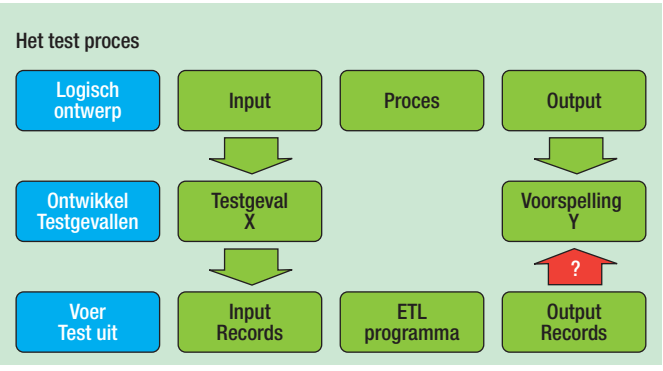
Kwaliteitsaspecten van logische testgevallen

De vraag kan worden gesteld waarom het opstellen van de testgraaf de gewenste werkwijze is om logische testgevallen af te leiden. Daarom wordt een nadere detaillering van het begrip 'kwaliteit' gegeven. Deze nadere detaillering wordt gedaan op basis van de publicaties van Cavano en McCall en van Boehm over aspecten van de kwaliteit van software-componenten, zie [2]. Hierin staat het begrip 'blijvende bruikbaarheid' centraal.

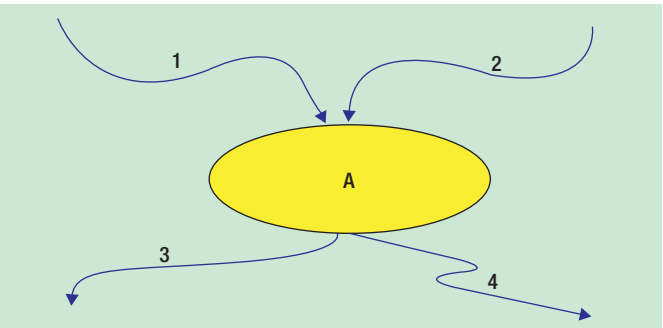
De blijvende bruikbaarheid van een product (in ons geval dus logische testgevallen) wordt bepaald door de huidige bruikbaarheid, de onderhoudbaarheid en de overdraagbaarheid, zie afbeelding 2.
De vraag is dus hoe het generatieproces van logische testgevallen er uit moet zien, hoe dat proces gedocumenteerd moet worden en hoe de logische testgevallen zelf beschreven kunnen worden, wil het voldoen aan de hierboven beschreven eisen. Dat werken we hierna uit. Eerst volgt wat theorie, dan een klein voorbeeld en uiteindelijk controleren we of de methodiek op elk onderdeel uit een detailontwerp kan worden toegepast.

Ontwerpen, bouwen en testen

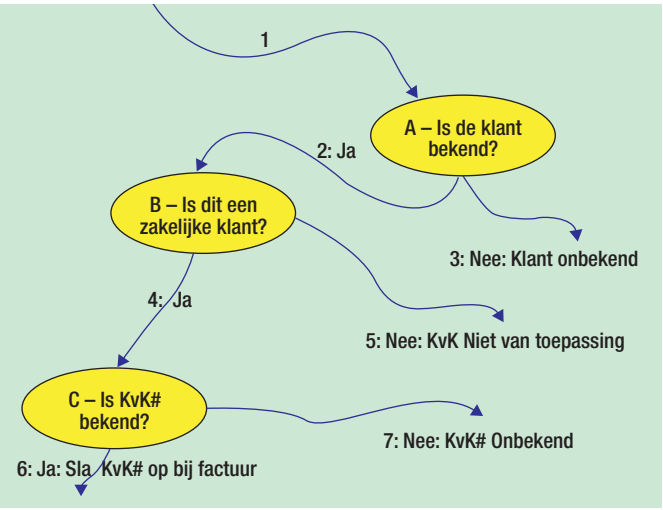
Het testen van software is een essentieel onderdeel van het systeemontwikkelingsproces. Het kan kortweg worden omschreven als het toetsen van het geleverde product tegen de specificaties voor dit product. Bij het bouwen en testen van ETL-programma's is dit niet anders.
Op basis van een detailontwerp worden een applicatie ontwikkeld en logische testgevallen opgesteld. De logische testgevallen worden omgezet naar technische testgevallen (tabellen met relevante gegevens) en een voorspelling van het resultaat. Vervolgens wordt de testomgeving met de technische testgevallen



Afbeelding 3: Samenhang tussen ontwerpen en testen.



Afbeelding 4: Het algemene voorkomen van een beslispunt met enkele voorliggende en volgende acties.



Afbeelding 5: Een keten van beslissingen en acties.

gevuld en wordt de applicatie met de fysieke testgevallen gevoed. Het fysieke resultaat wordt vergeleken met het verwachte resultaat, zie afbeelding 3.

In datawarehouse-omgevingen is het van nature geen haalbare 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. Als er sprake is van zeer grote hoeveelheden gegevens komt het accent eerder te liggen op een *stress of performance test* dan op een test van de functionaliteit. En veel gegevens verwerken betekent ook een langere doorlooptijd van elke test;
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.

Hoe dan een 'volledige en juist voldoende verzameling van logische testgevallen' uit het ontwerp af te leiden? Het detailontwerp 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 detailontwerp dient daarom ook geschikt te zijn voor het ondersteunen van het testproces. Het vervaardigen van een testset uit een detailontwerp wordt hierna uitgewerkt.

Beslispunten en acties

Elk onderdeel van een detailontwerp kan, gegeneraliseerd, worden beschouwd als:

1. hetzij de beschrijving van een beslissing – beslispunten;
2. hetzij de beschrijving van een uit te voeren handeling – acties;
3. een beschrijving van de gewenste volgorde – actiecombinaties.

		OK?	Invulling door de methodiek.
Bruikbaar	Volledig	Nee	Hierna werken we uit dat het Detailontwerp volledig is om te zetten in beslispunten, acties en actiecombinaties.
	Nauwkeurig	Ja	De specificatie van logische testgevallen is expliciet. De constructie van logische testgevallen uit actiecombinaties is expliciet. De constructie van actiecombinaties is expliciet.
	Consistent[1]	?	De aanpak mag zichzelf niet tegenspreken.
	Efficiënt	Ja	Drie beslispunten met twee uitkomsten zouden acht logische testgevallen kunnen opleveren als geen gebruik wordt gemaakt van de samenhang. Nu worden vier logische testgevallen geconstrueerd. De efficiëntie neemt toe bij een groter aantal actiecombinaties, doordat in één logisch testgeval meerdere actiecombinaties worden opgenomen.
	Toegankelijk	Ja	De visualisatie verduidelijkt de functionaliteit. Het statement 'deze beslispunten en acties worden getest' is via de testgraaf uitstekend te communiceren. De volgorde van afhandeling voegt zelfs iets toe. De uiteindelijke logische testgevallen zijn, door de verwoording, toegankelijk en overdraagbaar.
Onderhoudbaar	Testbaar	Ja	De constructie van de logische testgevallen is navolgbaar en eenduidig.
	Gestructureerd	Ja	Na de formulering van beslispunten en acties volgt een gestructureerd te formaliseren proces.
	Begrijpelijk	Ja	De gebruikte aanpak vereist geen voorkennis en heeft een korte leercurve.
	Wijzigbaar	Ja	Wijzigingen in het detailontwerp kunnen eenvoudig in de graaf worden ingepast en leiden tot een nieuw logisch testgeval.
	Flexibel [2]	?	Uitzonderingen kunnen zonder al te grote inbreuk op de structuur worden beschreven.
Overdraagbaar	Platformonafhankelijk	Ja	De geconstrueerde logische testgevallen refereren nergens naar hulpmiddelen.

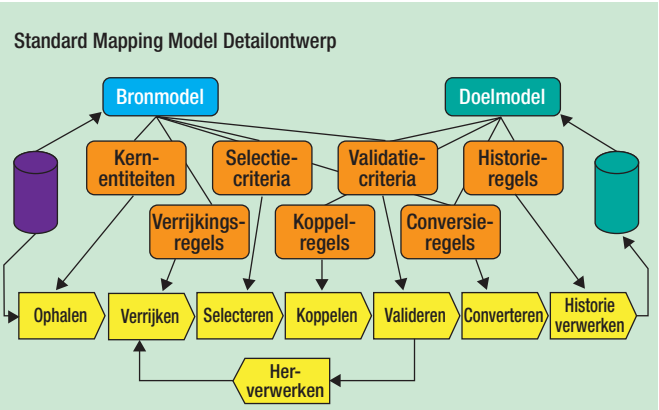
Afbeelding 6: Kwaliteitscriteria toegepast op de methode om logische testgevallen af te leiden.
(1. en 2. De werkgroep bekijkt nog hoe de criteria "Consistent" en "Flexibel" kunnen worden geoperationaliseerd.)

In afbeelding 4 (en in het vervolg van dit artikel) zijn acties genummerd en beslispunten voorzien van een letter. In de illustratie volgt na beslispunt A hetzij de actie 3 of 4. Uit het detailontwerp blijkt dat beslispunt A wordt bereikt na hetzij actie 1 of 2. Om dit deel van het detailontwerp goed te testen zijn dan de volgende actiecombinaties van belang:

- 1A3 – Na actie 1 wordt in A tot actie 3 besloten;
- 1A4 – Na actie 1 wordt in A tot actie 4 besloten;
- 2A3 – Na actie 2 wordt in A tot actie 3 besloten;
- 2A4 – Na actie 2 wordt in A tot actie 4 besloten.

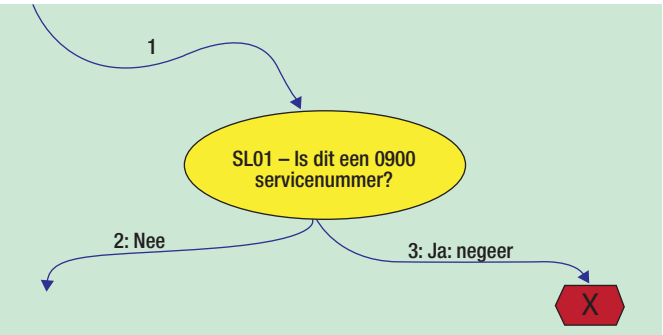
Een detailontwerp bevat natuurlijk een hele serie van beslispunten: door de actiecombinaties te combineren in één plaat ontstaat de testgraaf. In de testgraaf wordt de onderlinge samenhang toegankelijk en nauwkeurig gevisualiseerd. Genoeg theorie: Hoe ziet dat er dan uit? Voordat we een heel detailontwerp gaan uitwerken nemen we als voorbeeld een keten van beslispunten en acties zoals in afbeelding 5.
Afbeelding 5 zou voor zich moeten spreken, toch volgt een citaat uit het detailontwerp:
[...] Bepaal van facturen of de klant een registratie bij de Kamer van Koophandel heeft ten behoeve van de risicoclassificatie. Als de klant nog onbekend is, zet dan de verwijzing van de klant op onbekend. Als de klant bekend is en het is een zakelijke klant, haal dan het Kamer van Koophandelnummer op. Als het nummer niet gevonden kan worden zet het KvK-nummer dan op onbekend. [...]

De tester heeft dit stuk tekst omgezet naar drie beslispunten en zeven acties. Illustratie 5 geeft uitstekend inzicht in de gewenste functionaliteit. Om dit deel van het detailontwerp goed te testen zijn de volgende actiecombinaties van belang om te testen: 1A2; 1A3; 2B4; 2B5; 4C6; 4C7.
Deze actiecombinaties zijn samen te voegen tot logische testgevallen, door de actiecombinaties als dominostenen aan elkaar



Stap	Doel
Ophalen	Alle kandidaat records verzamelen uit het bronmodel
Verrijken	Alle kandidaat records met indirect gekoppelde attributen
Selecteren	Alle relevante kandidaten
Koppelen	Alle relevante kandidaten met verwijzende attributen naar het doelmodel
Valideren	Alle goedgekeurde kandidaten
Converteren	Alle goedgekeurde kandidaten met alle benodigde attributen voor het doelmodel

Afbeelding 7: De stappen van een ETL Detailontwerp.



Afbeelding 8: De selectie als beslissing en actiecombinatie.

Nr	Expressie	Conditie	Actie indien niet voldaan aan de conditie	Foutmeldings-tekst
VD09	Verrijking VR01	Geslaagd	Afwijzen	Geen extra info
VD01	GSPK.Datum	Is van het formaat "dd mon"	Afwijzen	Ongeldige datum
VD08	Koppeling met UURINDELING	Geslaagd	Afwijzen	Onbekende uurindeling

Afbeelding 9: Validaties van velden en de verrijkingen, koppelingen.

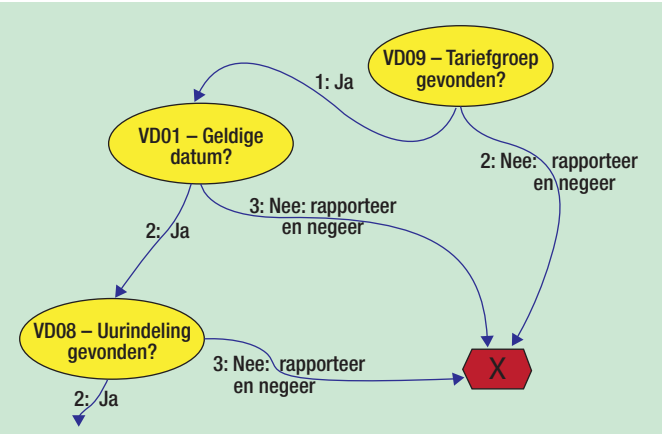
te klikken en van boven af naar beneden te werken. Een voorbeeld:

- 1A2 sluit aan op 2B4. 2B4 sluit aan op 4C6;
- 1A2B4C6 is dan een *logisch testgeval*, dat is te *verwoorden* als: Een factuur van een bekende klant waar het KvK-nummer van bekend is.

Elk van de drie resterende actiecombinaties vormt drie logische testgevallen op zich:

- 1A3 – Een factuur van een (nog) onbekende klant;
- 2B5 – Een factuur van een niet-zakelijke klant;
- 4C7 – Een factuur van een zakelijke klant waarvan het KvK nummer niet bekend is.

Vier logische testgevallen zijn samen nodig en juist voldoende om de functionaliteit voor 100 procent te dekken: 100 procent van de beslispunten; 100 procent van de acties; 100 procent van de actiecombinaties. Een logisch testgeval is bovendien eenvoudig te verwoorden in een goede Nederlandse zin, die door de opdrachtgever, ontwerper, bouwer, tester en beheerder te begrijpen is.



Afbeelding 10: De validaties VD09, VD01 en VD08 in samenhang.

Het is interessant de methodiek nu al te beoordelen en de kwaliteitscriteria af te lopen, om te bezien welke aspecten al aan bod zijn gekomen en welke nog te bespreken zijn. We herhalen in de tabel in afbeelding 6 de eerste twee kolommen uit afbeelding 2 en geven in de derde en vierde kolom aan of, ons inziens, aan de kwaliteitseis is voldaan. Zo ja dan volgt daarna de invulling.
We constateren dat aan het grootste deel van de criteria is voldaan. Het belangrijkste nog te controleren punt is de volledigheid: is elk detailontwerp met deze methodiek te vertalen in logische testgevallen? Laten we daarom eens bekijken hoe, als een detailontwerp is gestructureerd zoals beschreven in eerdere publicaties van ons in [1], daaruit eenvoudig een testgraaf ontstaat. Eerst brengen we de stappen van een ETL-detailontwerp in herinnering en vervolgens werken we per stap in het detailontwerp de verwachte vertaling naar actiecombinaties uit, zie afbeelding 7. Voor het praktische vervolg hebben we behoefte aan een voorbeeld. We gebruiken daarvoor een detailontwerp van een proces dat gegevens gereed zet voor het analyseren van belgedrag en belkosten van gebruikers van vaste en mobiele telefoontoestellen. De gesprekken naar 0900-nummers worden niet in deze tabel opgenomen. De gesprekken worden ingedeeld naar moment van de dag (dal of piek).

Ophalen en verrijken.

Het onderdeel *Ophalen* bevat de trigger-tabel waaruit de kandidaten geput worden en bevat dus geen beslispunten. Het deel *Verrijken* beschrijft de te leggen relaties tussen tabellen in het bronmodel. Het niet slagen van een verrijking wordt afgehandeld bij de stap *Validatie*. Dit deel zal dus geen beslispunten bevatten.

Selecteren.

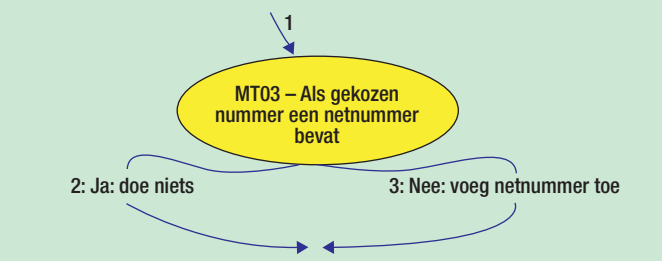
Dit onderdeel bevat de volgende beslissing:

Nr	Selectie
SL01	/* Is dit een 0900 servicenummer? */ Is GSPK.RUBRIEK <> "0900- Servicenummers"

Dit is te visualiseren zoals in afbeelding 8.

Nr	Expressie
MT03	Als in het nummer geen netnummer staat vul dat dan aan. ## Indien (in [Gekozen_Nummer] zit geen '-') ## dan GPKX.A_Netnummer+'-'+GSPK.Gekozen_Nummer ## anders GSPK.Gekozen_Nummer

Afbeelding 11: Een voorbeeld van een manipulatie.



Afbeelding 12: De actiecombinatie MT03.

Koppelen.

Koppelingen betreffen de te leggen relaties tussen de kandidaat-records en het doelmodel. Het niet slagen van een koppeling wordt afgehandeld bij de stap *Validatie*. Dit deel van het detailontwerp zal dus geen beslispunten bevatten.

Valideren.

Dit deel van het detailontwerp beschrijft de uit te voeren validaties plus de acties die worden ondernomen als niet aan een conditie wordt voldaan. In het detailontwerp staat beschreven dat de validaties in de genoemde volgorde moeten worden uitgevoerd. In het oorspronkelijke detailontwerp stonden negen beslissingen en acties. We bekijken VD09 omdat het een validatie uit de verrijkingsstap betreft, VD01 omdat het een 'gewone' domeinvalidatie betreft en VD08 omdat het de validatie van een koppeling is, zie afbeelding 9 en 10.

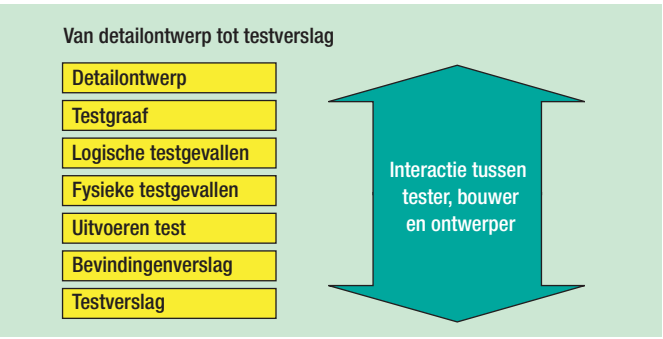
Manipulatie.

In de manipulatietabel kunnen kleine beslissingen worden genomen, zoals in afbeelding 11. En dit is te visualiseren zoals in afbeelding 12.

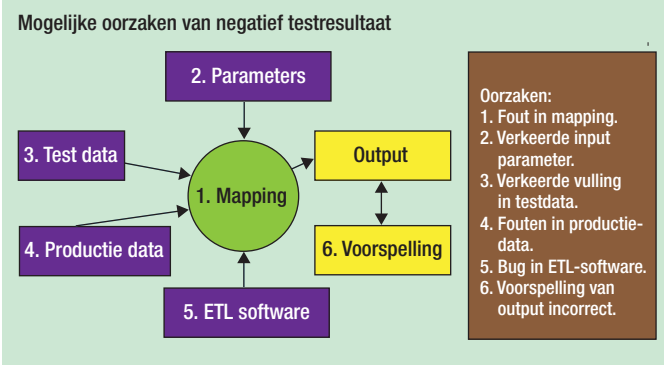
De vertaling van het detailontwerp zoals eerder voorgesteld, verloopt per onderdeel van het door ons voorgestelde detailontwerp vlot, eenduidig en volledig, waarmee ook aan een van de laatste kwaliteitseisen is voldaan. Dan resteert ons nog een 'formele' beschrijving van het proces. De methode voor het vaststellen van de logische testgevallen kent de volgende stappen.

Maak de testgraaf:

1. Zoek in het detailontwerp naar de beslispunten;
2. Bepaal per beslispunt de mogelijke keuzeopties;
3. Bepaal per keuzeoptie de verwachte actie;



Afbeelding 13: De gebruikelijke volgorde van uitvoering van het testwerk.



Afbeelding 14: Te controleren objecten in de omgeving van een te testen ETL-proces.

4. Maak een testgraaf met alle beslispunten en acties in samenhang;
5. Bespreek de testgraaf met de ontwerper, bouwer en beheerder.

Maak de logische testgevallen:

6. Stel een lijst van actiecombinaties op;
7. Bepaal de logische testgevallen door actiecombinaties aan te schakelen;
8. Verwoord de gevonden logische testgevallen;
9. Bespreek de logische testgevallen met de ontwerper, bouwer en beheerder.

Gedurende het proces is elke onduidelijkheid een verbetervoorstel voor het detailontwerp. Daarom bepleiten wij het opstellen van de grafen bij aanvang van de bouw. Opmerkingen zouden zelfs in het bevindingenverslag van de tester kunnen worden opgenomen.

Samenvatting

Het testen is een zeer uitdagend traject, omdat de tester veel onder controle dient te houden om één object, een nieuw applicatieonderdeel goed te kunnen beoordelen en er zeker van te zijn dat onverwachte resultaten te herleiden zijn naar de applicatie. Het is daarom van groot belang de verzameling van logische testgevallen goed te laten aansluiten bij het detailontwerp en te voorkomen dat een deel van het testwerk ook ter discussie komt te staan. De constructie van logische testgevallen met behulp van een testgraaf biedt in ruime mate houvast. De testgraaf zelf geeft de tester alle middelen anderen inzicht te geven in zijn werk ten bate van het project en het beheer.

Literatuur

1. *DB/M 6/2004, 6/2005 door Mark Zwijsen en 7/2005 door Mark Zwijsen en Rob Eveleens.*
2. *J.P. Cavano en J.A. McCall, A framework for the measurement of software quality (1978). Proceedings of the Software Quality and Assurance Workshop, 133-140.*

Rob Eveleens, Kees Molenaar, Yvonne Schurink, Mark Zwijsen

Rob Eveleens (rob.eveleens@atosorigin.com), Kees Molenaar, Yvonne Schurink, Mark Zwijsen (mark.zwijsen@atosorigin.com) zijn consultants bij Atos Origin.