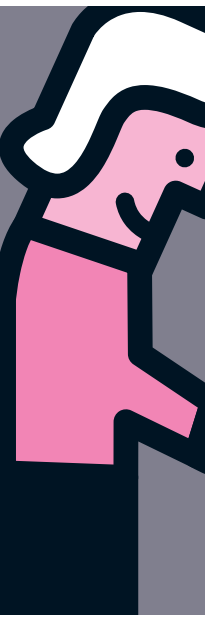

Testautomatisering levert enorme tijdsbesparingen op

Testen van BI anders dan traditioneel testen

Chiel Bleumink, Gabriel Felten en Jos van Rooyen

Slechte informatiekwaliteit kan vergaande consequenties hebben voor bedrijven of organisaties. Op alle niveaus kunnen verkeerde besluiten worden genomen als beschikbare informatie niet compleet of niet juist is. Hoe is dit te voorkomen? Er zijn enkele valkuilen bij de ontwikkeling van Business Intelligence-systemen waarmee rekening gehouden dient te worden, niet alleen in het ontwerp- en bouwproces maar ook in het testproces.

Dit artikel gaat in op de aanpak van Business Intelligence testen en adresseert in het bijzonder de specifieke aandachtspunten ten aanzien van het testen van het datawarehouse. De specifieke aspecten met betrekking tot teststrategie, testvoorbereiding, testuitvoering, testautomatisering en begroten worden behandeld.

Het testen van BI-oplossingen vereist een andere benadering dan conventionele testtrajecten. Er spelen specifieke product-risico's. Vaak zijn de te testen processen complex van aard en vindt testen plaats op database-niveau, zonder een beschikbare front-end. Dit zorgt ervoor dat er voor een BI-testtraject meer nodig is dan een standaard Risk and Requirement Based Testing (RRBT) benadering [1].

Een BI-testtraject verschilt van conventionele testtrajecten, in het bijzonder voor de zogenaamde ETL-processen en het datawarehouse. Vragen die spelen zijn: waar dient het zwaartepunt van het testtraject te liggen en waar juist niet; wat is de meest effectieve testaanpak; welke valkuilen kunnen optreden in dergelijke testtrajecten? [7] De focus ligt hierbij op de systeemtest.

Testaanpak en bepalen teststrategie

Een standaard testaanpak bestaat uit het opstellen van een teststrategie, samenstellen van een testplan, opstellen van een begroting, de testvoorbereiding en uiteindelijk de testuitvoering. Zoals ook bij andere typen systemen geldt is het lang niet altijd efficiënt of mogelijk alles te testen. Voor de samenstelling van de teststrategie kan gebruik worden gemaakt van de Risk and Requirement Based Testing methode [1]. Als eerste stap worden de zogenaamde productrisico's vastgesteld.

Voor de extractie kan men denken aan productrisico's, zoals het aantal te selecteren bronrecords dat niet klopt of ontbrekende velden die in het datawarehouse cruciaal zijn voor de business om de gewenste analyses uit te voeren.

Voor de transformatie zijn productrisico's denkbaar als verkeerde datatransformatie leidt tot foutieve database-vulling, misinterpretatie van data door verkeerde combinatie van gegevens en datatypes in bron en doel. Dit levert problemen op bij impliciete conversies.

Voor de load zijn als mogelijke productrisico's te noemen; draaitijden van de ETL-processen zijn te lang en beschikbare database-ruimte is te klein om alle data te verwerken.

In de ontwikkeling van de teststrategie moet naast de product-risico's ook rekening worden gehouden met aandachtspunten zoals de compleetheid van de data [2]. Hoe is vast te stellen dat alle data correct zijn overgenomen? Worden de transformatie-regels correct toegepast? Welke vierkantsvergelijkingen moeten worden gemaakt en wat zijn de eisen ten aanzien van performance en scaleability?

Voor genoemde onderdelen zal de prioriteit en diepgang van de test moeten worden bepaald. In de teststrategie kan voor een gelaagdheid worden gekozen. Daarbij zijn veel overeenkomsten te constateren met het testen van conversieprojecten [3]. Zoals ook bij het testen van conversies is het belangrijk dat de ETL-regels consistent, compleet en correct zijn. Een middel daartoe is statisch testen. Om na de realisatiefase de correctheid van de software te kunnen vaststellen moet er een geconditioneerde set met testdata samengesteld worden, die alle ETL-regels afdekt, daarmee aantonend dat de regels correct zijn ingebouwd. Dit zal een iteratief proces zijn. Nadat vastgesteld is dat de ETL-regels correct zijn gebouwd, moet de test herhaald worden met behulp van productiedata om performance en scaleability vast te stellen.

Na het vaststellen van de teststrategie kan de testvoorbereiding worden opgestart. De diverse onderdelen van de teststrategie wordt uitgezet onder de testers. Een hulpmiddel daarvoor kan de clusterkaart zijn [1].

Testvoorbereiding

De testvoorbereiding bestaat uit verschillende onderdelen: het reviewen van het functioneel ontwerp, de analysefase en het genereren van de testdata. Voor het reviewen van het functioneel ontwerp zijn diverse technieken beschikbaar; zoals reviews, inspecties of een *walkthrough*. De keuze is afhankelijk van de doelstellingen. Wat zijn de specifieke aandachtspunten bij het reviewen van een functioneel ontwerp?

De opdrachtgever heeft meestal een beeld voor ogen op applicatieniveau. Daar weet hij exact wat hij wil zien met de te bouwen applicatie. Vanuit dit perspectief is de stap naar requirements voor een ETL-proces een lastige. Want welke velden in de database corresponderen met datgene wat de opdrachtgever straks op zijn scherm wil zien? En welke transformatieregels zijn er dan op welke velden van toepassing? En het kan ook zo zijn dat de opdrachtgever gegevens wil zien die alleen indirect af te leiden zijn.

Het is belangrijk om tijdens een review mee te denken met de functioneel ontwerper. Stap uit de box die tester heet en kijk verder: niet alleen naar de testbaarheid van een ontwerp, maar denk ook na over de beschreven oplossingen. Stel de vraag: zorgen deze requirements er inderdaad voor dat straks de business users krijgen wat ze willen?

Na de review-fase kan gestart worden met de daadwerkelijke testanalyse. Deze bestaat uit drie stappen:

1. Bepalen van alle beslismomenten in het functioneel ontwerp;
2. Uitwerken van testcondities en testgevallen aan de hand van de gekozen testdiepgang;
3. Creëren van de benodigde testdata en bijbehorende testvoorspellingen.

Leg hierbij de nadruk op de tabeldefinities, bronselecties, een-op-een overnames, transformatievelden, verrijkingsvelden en data-integriteit. Deze aandachtsgebieden vormen namelijk de essentie van een ETL-proces. Definieer in beginsel dus geen testgevallen die bijvoorbeeld het maximaal aantal karakters in een veld testen of die de mogelijke karakters testen welke een veld wel of niet mag bevatten. Deze testen zijn meer bestemd voor het stadium waarin de betreffende front-end applicatie getest wordt. Een aantal specifieke aandachtsgebieden staat vermeld in het kader.

Bij het samenstellen van de testgevallen moet met nog een aantal zaken rekening worden gehouden [6,7]. Uiteraard is een en ander afhankelijk van de teststrategie. Denk bijvoorbeeld aan de rudimentaire volgorde van de testgevallen. Iedere keuze-optie leidt tot minstens twee testgevallen, elke mogelijke combinatie van binnenkomende en uitgaande acties rond een keuzemoment moet in een logisch testgeval voorkomen. Bij het ontwikkelen van testgevallen moeten de hoofdroutes altijd uitgewerkt worden in testgevallen, eventueel verrijkt met externe data. In een speciaal cluster kunnen testgevallen worden ontwikkeld voor het detecteren van combinatiefouten of vierkantsvergelijkingen.

Voor het opstellen van de testgevallen kunnen testtechnieken worden toegepast. De traditionele technieken liggen niet voor de hand, omdat technieken als grenswaarde-analyse en equivalentieklassen vooral voor online systemen van toepassing zijn. Voor het testen van datawarehouse-toepassingen zijn technieken als hash totals, audit trail en vierkantstellingen van toepassing.

De gedefinieerde testgevallen moeten uitgewerkt worden naar testdata. Hiervoor zijn tools ontwikkeld die de logische en fysieke testgevallen vertalen naar datastrings die in de testomgeving ingelezen kunnen worden [4]. Het grote voordeel van deze aanpak is dat de uitvoervoorspelling altijd herbruikbaar is. Immers de set aan testgevallen en de daarbij behorende testdata zijn altijd hetzelfde [5]. Voor het testen van de load-processen moet er uiteraard wel een representatieve hoeveelheid data zijn. Wordt gebruik gemaakt van een selectie uit de bronsystemen dan is de kans aanwezig dat de uitvoervoorspelling niet meer klopt door de wijzigingen in het bronsysteem.

Activiteit	Handmatige acties	Tijd*	Te automatiseren?
1. Review Functioneel Ontwerp.	Functioneel ontwerp bestuderen en opmerkingen verwerken.	L	N
2. Bepalen testcondities en testgevallen.	Aan de hand van het FO beschrijven van testcondities en testgevallen.	M	N
3. Bepalen van testdata nodig in brondatabase.	Aan de hand van testcondities en testgevallen opstellen en vastleggen van benodigde testdata.	M	N
4. Bepalen van testvoorspellingen.	Aan de hand van testcondities en testgevallen opstellen en vastleggen van testvoorspellingen.	M	N
5. Uitvoeren intake-test.	Query's opbouwen om tabeldefinities uit te lezen. Handmatige controle van deze tabeldefinities tegen functioneel ontwerp.	H	J
6. Toevoegen van testdata in brondatabase.	Insert-query's opbouwen om testrecords toe te kunnen voegen in brondatabase.	H	J
7. Controle van testvoorspelling met werkelijke inhoud datawarehouse.	Query's opbouwen om datawarehouse uit te lezen. Vervolgens handmatige controle per veld.	H	J
* Maat voor tijdsbesteding bij handmatig testen van ETL-processen: H = Hoog; M = Middel; L = Laag			

Afbeelding 1: Overzicht activiteiten.

Wanneer alle testgevallen en testrecords genummerd zijn, is het mogelijk om ze met diezelfde nummers te relateren aan de bijbehorende requirements uit het functioneel ontwerp. Op deze wijze wordt volledige traceability gecreëerd. Dit geeft naast een goed overzicht ook een snel inzicht in de mogelijke impact van functionele wijzigingen.

Testuitvoer

Na de analysefase volgt de fase testuitvoer. Deze verloopt in vijf stappen: 1. Uitvoeren intake test; 2. Toevoegen van testdata; 3. Draaien van de ETL-processen; 4. Controle van testvoorspellingen; 5. Uitvoeren van extra controles.

Het doel van de intake-test is te kijken naar de nieuw aangemaakte tabellen in het datawarehouse. Zijn de juiste tabeldefinities aanwezig? Draaien de ETL-processen? Immers, wanneer deze al niet correct zijn, heeft het geen zin om met testen door te gaan. De volgende activiteit is het toevoegen van de vooraf gecreëerde testdata aan de brondatabase. Zorg er ook voor om database-vervuiling te voorkomen, dat toegevoegde testdata weer snel en gemakkelijk te verwijderen zijn. Dit kan met behulp van een tool of door middel van handgemaakte delete query's. De inhoud van de database is dan na de test weer terug te brengen in de originele staat.

Na het toevoegen van testdata en het uitvoeren van het te testen ETL-proces kunnen de testvoorspellingen worden gecontroleerd. Een goed tool kan het vergelijken van de testvoorspellingen met de records in het datawarehouse automatiseren. Dit scheelt enorm veel tijd en bovendien is het nauwkeuriger. Een tool ziet namelijk niet zomaar velden over het hoofd – iets dat met handmatige controles wel snel kan gebeuren.

Naast het controleren van de testvoorspellingen kunnen nog extra controles worden uitgevoerd. Voorwaarde hiervoor is wel dat de betreffende brondatabase naast de zelf toegevoegde testdata, voldoende productiegelijkende data bevat. Na het draaien van het ETL-proces kunnen door middel van query's analyses op bron en doel worden losgelaten. Om een idee te geven van de mogelijke extra controles is hierna een voorbeeld uitgewerkt.

Analyse op de bron-database laat zien dat deze voor 23 procent is gevuld met product X, voor 56 procent met product Y en voor 21 procent met product Z. Deze variatie tussen X, Y en Z zou men ook in het datawarehouse moeten terugzien. De percentages zullen in de praktijk nooit exact overeenkomen wanneer de brondatabase niet in zijn geheel wordt overgenomen. Echter, zeer grote afwijkingen op deze percentages geven wel aan dat er nader onderzoek kan worden gedaan om te bepalen of deze afwijking al dan niet terecht is.

Testautomatisering

Met de moderne ontwikkelomgevingen is een datawarehouse snel gebouwd maar nog niet getest. De tabel in afbeelding 1 geeft een overzicht van activiteiten die wel of niet geautomatiseerd kunnen worden getest. Voor het automatiseren zijn diverse tools op de markt beschikbaar. Daarnaast zijn add-on's ontstaan op testmethoden als TestFrame, welke alle benodigde functionaliteit bevat om de stappen in de tabel te automatiseren. Deze add-on's werken via een invoegtoepassing in Microsoft Excel [4].

Begroten van ETL-processen

Kijkend in de literatuur bestaat er niet echt een methode voor

Te begroten object	Wide Band Delphi	Test omvang	Ervaringscijfers
Extractie:			
Extractie	X	X	X
Uitval			X
Keten	X		X
Regressietest		X	X
Crosschecks			
Transformatie:			
Transformatie	X	X	X
Crosschecks	X		X
Load:			
Load-proces	X		X
Rapportages		X	
Overig:			
Load & Performance	X		X
Security	X		X
Usability	X		X
Testdata	X		X
Queryproces	X		X

Afbeelding 2: Begrotingstechniek per onderwerp.

het begroten van dergelijke testprojecten. De vraag is natuurlijk hoe dat komt; een goede vraag zonder een eenduidig antwoord. Vandaar dat in deze paragraaf niet een methode wordt aangereikt, maar een combinatie van gangbare technieken.
Welke aspecten dienen begroot te worden? We beginnen met de functionaliteit. Vanuit het extractieproces moeten zaken als de extractie zelf, het doorlopen van de keten, begroten van de uitval (data die niet door de extractieregels komen), begroten van *cross checks* met andere onderdelen van het datawarehouse om consistentie van de data vast te stellen, begroot worden. Als er incrementeel ontwikkeld wordt zal ook de regressietest in de begroting meegenomen moeten worden.
Het transformatieproces vraagt om een begroting voor alle velden die getransformeerd worden. Vaak zijn dit de requirements die zijn opgenomen in het functioneel ontwerp. Ook hier dienen cross checks met andere onderdelen van het datawarehouse begroot te worden om consistentie van de data vast te stellen. Het load-proces is grotendeels een geautomatiseerd proces. Ten aanzien van de begroting moet de query begroot worden waarmee het load-proces wordt gestart met daarbij de doorlooptijd. Als onderdeel van het load-proces worden ook de rapporten gezien die automatisch gegenereerd worden vanuit het load-proces.
Naast de functionele aspecten zijn, afhankelijk van de requirements, ook niet-functionele aspecten een punt van aandacht in de begroting. In het voorgaande zijn de aspecten beschreven die men moet begroten. Wat is nu de beste techniek om de diverse aspecten te begroten? Er zijn momenteel drie technieken [8] voorhanden die het best toepasbaar zijn. Het betreft:
1. De Wide Band Delphi methode;
2. Begroten op basis van test omvang;
3. Ervaringscijfers, (bijvoorbeeld Test Effort Estimation Model).
Welke technieken het beste voor welk onderdeel toepasbaar zijn, staat opgenomen in de tabel in afbeelding 2. Een aandachtspunt bij het begroten van ETL-processen is dat het niet mogelijk is om standaard verhoudingscijfers (FO, Bouw, Test) te gebruiken. De reden is dat door de moderne ontwikkel-tools het relatief simpel is om applicaties als ETL-processen te programmeren. Echter, de benodigde testinspanning neemt niet af. Neemt men een standaard verhoudingscijfer dan komt men altijd tijd te kort, of de testomvang moet teruggebracht worden naar de hoeveelheid beschikbare tijd. Dit gaat uiteraard gepaard met een flink groter (meestal te groot) risico lopen van kwaliteitsverlies. Testen nemen bij ETL-processen om bovenstaande reden vaak meer dan de helft van de beschikbare projecttijd in beslag.

Conclusie

Slechte informatiekwaliteit heeft vergaande consequenties, vooral in de gevallen waarin bedrijven grotendeels afhankelijk zijn van Business Intelligence ter ondersteuning van het uitvoeren van hun kernactiviteiten. De gevolgen van inferieure informatiekwaliteit variëren van imagoverlies tot onnodige hoge kostenposten.

Aandachtsgebieden ETL-testen

Tabeldefinities: voldoen de aangemaakte tabellen in het datawarehouse aan de gestelde eisen? Denk aan veldtype, veldlengte, primary keys, nullable of not nullable, enzovoort.
Bronselecties: welke gegevens worden wel en welke worden niet overgenomen?
Een-op-een overnames: velden waarvoor geen transformatieregel geldt.
Transformatievelden: velden waarvoor wel een transformatieregel geldt.
Verrijkingsvelden: velden uit andere brontabellen die worden overgenomen om een record in de doeltabel aan te vullen.
Data-integriteit: richt zich op relaties tussen gegevens over meerdere tabellen heen. Houdt het ETL-proces rekening met de regels die hiervoor volgens het ontwerp zijn opgesteld?

Het testen van ETL-oplossingen onderscheidt zich in meerdere opzichten van het testen van conventionele systemen. Er zijn specifieke productrisico's van toepassing. De te ontwikkelen testgevallen moeten andere situaties afdekken dan bij traditioneel testen. ETL-testen vindt vaak plaats op database-niveau; het gebruik van testdata is cruciaal. Kijk kritisch naar de beschikbare brondata en de kwaliteit ervan. Als er geen data beschikbaar zijn betekent dit dat meer tijd begroot moet worden voor het genereren van testdata. Testautomatisering levert een enorme tijdsbesparing op voor zowel de testanalyse als de testuitvoer. Het begroten van ETL-testtrajecten is lastig. Het werken met standaard verhoudingscijfers (FO, Bouw, Test) gaat vaak niet op, vanwege intelligente ontwikkel-tools die gebruikt worden voor het bouwen van ETL-oplossingen [7].

Literatuur

1. *Succesvol testmanagement; een integratie-aanpak, Burgt & Pinkster, 2003.*
2. *Strategies for testing datawarehouse applications, Theobald, DMR review, June 2007.*
3. *BCS, Non functional testing standard, 2002.*
4. *Testen van Business Intelligence oplossingen, Koster M., Testnet thema-avond, november 2007.*
5. *Modelgedreven ontwerp van ETL-functies, Zwijsen & Eveleens, Database Magazine, november 2005.*
6. *Testing solutions for datawarehousing, Koomen T., Eurostar 2003.*
7. *Testen van datakwaliteit, Bouman E., Testnet thema avond maart 2005.*
8. *Syllabus ISEB practitioner.*

Chiel Bleumink, Gabriel Felten en Jos van Rooyen

Chiel Bleumink is testconsultant.

Gabriel Felten is senior testconsultant bij Bartosz ICT BV.

Jos van Rooyen (jos.van.rooyen@bartosz.nl) is senior testadviseur bij Bartosz ICT BV.

Met dank aan Maurice Koster en Rene Born.