Controleren datakwaliteit wordt bereikbaar voor alle bedrijfsprocessen

Data Quality meets SOA

Walter Eichhorn

Al lang voordat de analisten van Gartner met het begrip Service Oriented Architecture kwamen waren Data Quality-functies als service beschikbaar in Unix-, Windows- en Linux-omgevingen. Doorslaggevende reden voor het 'vooroplopen' van Data Quality was hoofdzakelijk van technische aard. Niettemin leidt de implementatie van een servicegeoriënteerde architectuur tot nieuwe en aangepaste eisen aan Data Quality-services. Daardoor nemen de mogelijkheden en het praktisch nut alleen maar toe.

Een klassieke toepassing van Data Quality is de controle van postadressen aan de hand van referentiegegevens, zoals straat- en plaatsnamen en de afhankelijkheden tussen postcodes enerzijds en plaatsen, straten en huisnummers anderzijds. Anders dan bij een gewone database moet het juiste adres ook worden gevonden wanneer de ingevoerde gegevens onvolledig zijn en schrijf- of luisterfouten bevatten. Doel is een hoge mate van precisie, zodat zoveel mogelijk foutieve adressen automatisch worden gecorrigeerd.

Een andere toepassing is het opsporen van doublures in het eigen bestand. Ook nu is het doel om ondanks onvolledige, afwijkende of foutieve invoer – of het nu een bedrijfsobject, een zakenpartner, een product of een *sales opportunity* is – snel en trefzeker tot identificatie te komen. Dit vereenvoudigt het zoeken en verhoogt daarmee de productiviteit van de gebruikers. Zo wordt ook de registratie van doublures voorkomen, dat wil zeggen dubbele entry's die betrekking hebben op hetzelfde object in de reële wereld. Het beoogde resultaat is een database die een consequente, volledige en eenduidige afspiegeling is van reëel bestaande objecten.

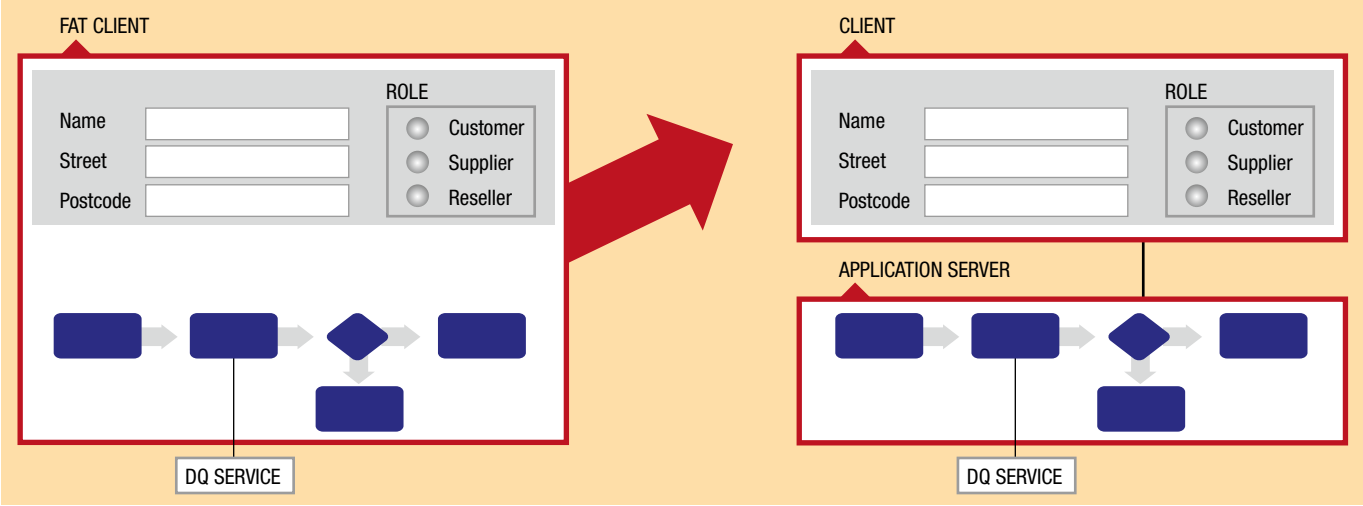
Voorkomen is altijd beter

Een belangrijk doel in het kader van verbetering van de datakwaliteit is eerst en vooral te voorkomen dat onvolledige of foutieve gegevens in de database terechtkomen. Eventuele onjuistheden zouden dus al bij het invoeren onderkend en vervolgens automatisch of handmatig (nadat de gebruiker van feedback is voorzien) gecorrigeerd moeten worden. Speciale zoekindexen zorgen ervoor dat een zoekopdracht in een gegevensbestand met één tot honderd miljoen gegevensrecords ook bij een afwijkende schrijfwijze in de regel slechts een fractie van een seconde vergt. Dergelijke responstijden vereisen echter een intelligente methode van indexcaching, wat uitermate efficiënt kan worden gerealiseerd door de software te implementeren als een centrale service die via een eigen server ter beschikking wordt gesteld.

Behalve de responstijd speelt integratie in de meest uiteenlopende omgevingen van oudsher een belangrijke rol bij Data Quality-services. Voor integratie waren twee dingen belangrijk: ontkoppeling van de Data Quality-service en de gebruiker van die service, en toepassing van een client/server-protocol. Integratiefunctionaliteit voor Data Quality werd zodoende, althans voor de meer veeleisende toepassingen, al vóór de 'uitvinding' van servicegeoriënteerde architecturen als service beschikbaar gesteld en gebruikt. Zo kon worden voldaan aan de hoge eisen die aan responstijden werden gesteld en de beschikbaarheid van functies in de meest uiteenlopende omgevingen kon worden gewaarborgd.

Naar standaardisatie en onafhankelijkheid

De typerende architectuur uit de beginjaren van de client/server-systemen bestond uit een database die niet alleen de opslag van transactie- en basisgegevens mogelijk maakte, maar ook asynchrone communicatie tussen de verschillende systeemcomponenten. Hierdoor was er sprake van ontkoppeling van deze componenten. Berichten (messages) werden naar de database verzonden en daar gelezen. Dit hield uiteraard in dat de desbetreffende tabel regelmatig moest worden gepolst om vast te stellen of nieuwe, nog niet afgehandelde berichten waren ontvangen. De *business logic* was voornamelijk op 'fat clients' geïmplementeerd, zie afbeelding 1. Eventuele batchverwerking werd afgehandeld via achtergrondprocessen met toegang tot de database. Interactieve functies voor het controleren van adressen of voor het herkennen van doublures tijdens het invoerproces waren in de grafische bovenlaag geïntegreerd. Doorgaans werden deze functies geactiveerd via een merkgebonden interface.



Afbeelding 1: Intrede gelaagde architectuur.

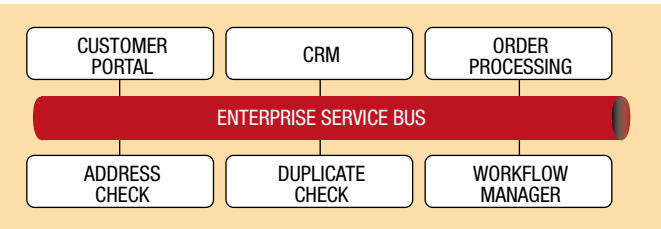
Specificaties zoals DCE en CORBA, die de standaardisatie van interfaces voor de communicatie tussen gedistribueerde componenten beoogden, waren veroordeeld tot een bestaan in de marge. Deze situatie is in de afgelopen jaren drastisch veranderd, vooral door de ontwikkeling van JEE-standaards (Java Enterprise Edition) en de beschikbaarheid van krachtige implementaties van die standaarden, in de vorm van zowel commer-ci-eel product als 'open source'-oplossing.
Microsoft volgde met de ontwikkeling van .NET als taalonafhankelijk platform en de .NET Enterprise Services voor Windows-omgevingen. Onafhankelijk van het gekozen platform (JEE, .NET) was nu krachtige infrastructuursoftware beschikbaar om de business logic grotendeels over te hevelen van de presentatielaag naar een eigen laag op de server. Daarmee veranderden ook de eisen die aan Data Quality-services werden gesteld, omdat deze nu hoofdzakelijk vanuit de business logic op de server werden benaderd. Eenvoudige integratie op de applicatieserver, hetzij op basis van een JEE- of een .NET-architectuur, trad nu op de voorgrond.

SOAP aan de basis van SOA

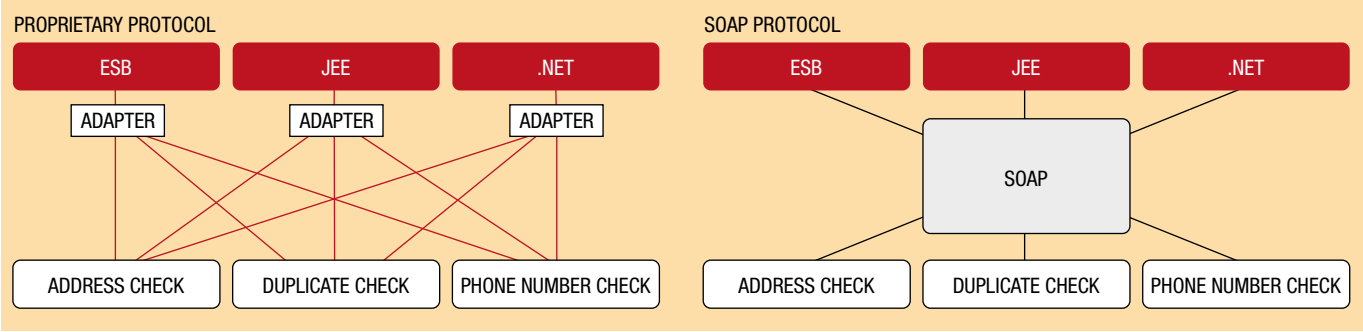
Echt effectief wordt een servicegeoriënteerde architectuur pas door toepassing van een standaardprotocol voor het beschikbaar stellen en gebruiken van services. Daarin werd voorzien met het webservice protocol SOAP. Dit protocol wordt door vrijwel alle middleware- en infrastructuurcomponenten ondersteund en maakt daardoor interoperabiliteit mogelijk tussen serviceaanbieders, middleware en servicegebruikers.
Daarmee zijn de connectoren overbodig geworden die nodig waren voor het gebruik van merkgebonden protocollen in merkgebonden middleware. En daarmee is ook de basis gelegd voor de ontwikkeling van zeer krachtige middlewarecomponenten. Voorbeelden hiervan zijn de Enterprise Service Bus (ESB) die een losse koppeling mogelijk maakt tussen verschillende componenten met de nadruk op het routeren van berichten, en *engines* voor het direct uitvoeren van *workflows* die in een Business Process-taal zijn gedefinieerd (BPEL). Zie afbeelding 2. Op SOAP gebaseerde webservices hebben zich zo ontwikkeld tot het centrale medium om interactieve Data Quality-services beschikbaar te stellen in moderne enterprise-architecturen. Daarbij gaat het voornamelijk om services volgens het Request/Response-patroon: de gebruiker van de service dient een verzoek in, bijvoorbeeld 'valideer het opgegeven adres', en de service reageert meteen met een bevestiging, een correctievoorstel of een selectie van mogelijke, correcte adressen. Deze werkwijze sluit goed aan op het interactieve karakter van het valideringsproces dat de gebruiker direct bij de registratie van een bedrijfsobject dient te ondersteunen en in geval van problemen oplossingen moet aanreiken. Deze functionaliteit is echter niet meer apart op de presentatielaag geïmplementeerd, maar maakt in alle gevallen deel uit van een bovenliggend bedrijfsproces, bijvoorbeeld het bestelproces in een e-businesstoepassing of het proces voor het converteren en kwalificeren van *leads* in een CRM-toepassing. De samenhang tussen bedrijfsprocessen en Data Quality-functies is meteen zichtbaar en daarmee ook de bijdrage die deze functies leveren aan het succes van het desbetreffende proces.

In de praktijk

De klanten van het Franse telecombedrijf Orange kunnen via diverse kanalen contact opnemen met het bedrijf. Ze kunnen de webportal van Orange bezoeken, ze kunnen het call center bellen of ze kunnen naar de telefoonwinkel gaan van een van de Orange-dealers. Ongeacht welk kanaal de klant kiest, de kwali-



Afbeelding 2: Enterprise Service Bus.



Afbeelding 3: De intrede van SOAP.

teit van het desbetreffende proces moet te allen tijde gewaarborgd zijn. Een belangrijk aspect van die proceskwaliteit is zekerheid over de juiste klantadressen.
De open source Java-applicatieserver JOnAS vormt de technische basis. Deze JEE-server heeft een centrale rol en stelt de services beschikbaar die nodig zijn om de bedrijfsprocessen van Orange te implementeren – waaronder ook de Data Quality-service van Uniserv voor het valideren, structureren en standaardiseren van klantadressen. Door deze servicegeoriënteerde benadering zijn voor verschillende processen en kanalen dezelfde services beschikbaar. Of het nu gaat om het invoeren van een nieuwe klant of het wijzigen van het adres van een bestaande klant, en ongeacht via welk kanaal dat proces in gang wordt gezet, de onderliggende servicegeoriënteerde architectuur zorgt ervoor dat de processen steeds op dezelfde manier verlopen en toegang hebben tot dezelfde services. Zo kan ondernemingsbreed een hoge, uniforme kwaliteitsstandaard voor adresgegevens worden gewaarborgd.

Lichtgewicht: REST

Het op SOAP gebaseerde protocol is ideaal voor het integreren van services in gangbare enterprise-middleware, maar er zijn toepassingen waar alternatieven meer op hun plaats zijn, bijvoorbeeld RESTful Services. Dat is vooral het geval wanneer Data Quality-services direct toegankelijk moeten zijn op de – op HTML/AJAX gebaseerde – presentatielaag. Een kenmerkend scenario in dit geval is invoerondersteuning, waarbij gedeeltelijk ingevoerde informatie automatisch wordt gecompleteerd of waarbij op basis van gedeeltelijke invoer suggesties ter completering worden aangereikt. Invoerondersteuning hoort van nature op de presentatielaag thuis en stelt duidelijk hogere eisen aan de responstijd omdat deze ondersteuning tijdens het invoerproces vaker – in de regel na elk ingevoerd teken – wordt aangeroepen. Overigens zijn voor het invoeren van adressen dezelfde services beschikbaar als voor het valideren van adressen.
Vanwege de overhead die met het SOAP-protocol is gemoeid, zijn op SOAP gebaseerde webservices niet altijd geschikt voor dit toepassingsscenario. RESTful-webservices zijn in vergelijking met webservices op basis van SOAP uitgesproken slank. De aanroep vindt plaats met behulp van het HTTP-protocol en aansluitend worden de aanroepargumenten in de URL gecodeerd. In het geval van een JavaScript-aanroep wordt het resultaat idealiter in JSON-formaat geleverd; dit levert een JavaScript-code op die door de JavaScript-interpreter van de browser direct kan worden geïnterpreteerd, waardoor het resultaat van de aanroep direct als JavaScript-object beschikbaar kan worden gesteld. Services die op deze wijze gestalte krijgen, zijn ideaal toe te passen in Web 2.0-applicaties.

SOAP – de fijne verschillen

Ook wanneer een merkgebonden interface wordt aangepast aan een communicatieproces op basis van SOAP, is er sprake van een leercurve. De gegevenspakketten die bij communicatie op basis van SOAP worden uitgewisseld, worden beschreven in de metataal WSDL (Web Service Description Language).
Eerst moet bij de ontwikkeling van de WSDL zorgvuldig erop worden toegezien dat de gebruikte datatypen door alle doeltalen en -systemen worden ondersteund waarbinnen de service wordt gebruikt. Bij een puur bedrijfsintern ontwikkelingsproces met een homogene software-infrastructuur, bijvoorbeeld JEE of .NET, is dit aspect minder kritiek. Gaat het echter om een Data Quality-service die in de meest uiteenlopende omgevingen gebruikt moet kunnen worden, waarvan sommige op voorhand mogelijk niet eens bekend zijn, dan is dat criterium van essentieel belang.
Bovendien biedt de SOAP-specificatie twee mogelijkheden om de koppeling tussen de gegevenspakketten in XML en de *constructs* van de programmeertaal die het pakket beschikbaar stelt of interpreteert, in WSDL te definiëren. De zogeheten 'rpc style' komt, zoals het acroniem al aangeeft, overeen met de klassieke Remote Procedure Call (RPC) en modelleert bewerkingen als *method calls* die niet verschillen van lokale aanroepen. Method calls zijn ideaal wanneer de webservice wordt aangeroepen vanuit een objectgeoriënteerde programmeertaal zoals Java of C#.
De zogenaamde 'document style' is juist meer geschikt voor het modelleren van complexe inhoud die als XML document met een eigen XML schema wordt weergegeven. Zo is enerzijds met standaard XML functies validering mogelijk aan de hand van het XML schema, anderzijds kan het resulterende document met standaard XML functies of met overeenkomstige frameworks verder verwerkt, getransformeerd of voor de presentatielaag geschikt worden gemaakt. Beide varianten hebben voor- en

nadelen. Services die pretenderen vanuit elke willekeurige context aangeroepen te kunnen worden, moeten mogelijk zelfs beide varianten ter beschikking stellen.
Webservices zijn van nature toestandloos (stateless). Dit betekent dat in het ideale geval geen deelresultaten maar het volledige resultaat van de aanroep worden geleverd, en dat twee opeenvolgende aanroepen volstrekt onafhankelijk van elkaar zijn.
Webservices moeten schaalbaar zijn; voorwaarde daarvoor is de hiervoor beschreven staat van toestandloosheid. Daarnaast dient de webservice bij voorkeur geen beroep te doen op *global resources*. Mochten deze toch nodig zijn, dan moeten het beheer en de synchronisatie van de toegang tot deze resources allesbehalve ad hoc worden geïmplementeerd voor de desbetreffende webservice. In plaats daarvan moeten *resource pools* worden ingezet, zoals geboden door de meeste applicatieservers en die ook als 'open source'-extensie bestaan. Het resultaat is een effectieve en configureerbare vorm van *pooling*.
Juist de laatste twee punten maken doorgaans een gedeeltelijk herontwerp noodzakelijk wanneer bestaande functionaliteit in de vorm van een webservice wordt aangeboden.

Geen onderscheid

De beschikbaarstelling van softwareservices en applicaties via Internet, ook wel aangeduid met Software on Demand, Software as a Service en Cloud Computing, is een onderwerp dat voortdurend aan belang wint. Vaak wordt het fundamentele en diepgaande onderscheid tussen lokaal geïnstalleerde software ('on premise') en via het Internet gebruikte software ('on demand') aangehaald. Vanuit SOA-perspectief bestaat dat onderscheid niet, want in de context van een servicegeoriënteerde architectuur maakt het niet uit hoe een service beschikbaar wordt gesteld. Juist op het gebied van Data Quality-services, die op basis van referentiegegevens validaties en correcties uitvoeren, biedt een via Internet aangeboden service – als alternatief voor een lokaal geïnstalleerde server – interessante, nieuwe toepassingsmogelijkheden.
De referentiegegevens waarvan de service gebruik maakt, moeten uiteraard regelmatig worden geactualiseerd. Dat betekent zowel een regelmatig terugkerende, handmatige inspanning als regelmatige betaling van abonnementsgeld aan de leverancier van de gegevens. Gaat het om de postcodes van meerdere landen, dan gelden deze inspanning en kostenpost voor elk afzonderlijk land waarvoor adrescontroles worden uitgevoerd. De toepassing van dergelijke oplossingen is daardoor alleen interessant bij grotere hoeveelheden gegevens. Deze beperking kan worden omzeild door de service beschikbaar te stellen als Software as a Service (SaaS), waarbij uitsluitend wordt betaald voor uitgevoerde transacties. Door toepassing binnen een servicegeoriënteerde architectuur kunnen lokaal geïmplementeerde en via Internet aangeboden services naar believen worden gecombineerd of elkaars plaats innemen. Zo kan voor iedere gebruiker de optimale oplossing worden gevonden, die bovendien kan worden aangepast aan veranderende omstandigheden.

Conclusies

Op basis van het voorgaande komen we tot de volgende vragen en antwoorden. Ten eerste, aan welke eisen dienen Data Quality-functies te voldoen die in een SOA worden ingezet?

- De functies moeten via SOAP als service kunnen worden benaderd. Zo is de integratie gewaarborgd in vrijwel alle infrastructuren voor een servicegeoriënteerde implementatie van bedrijfsprocessen. Wel dient nauwkeurig te worden toegezien op de juiste *mapping* van de XML-elementen van de servicebeschrijving en de overeenkomstige constructs van de desbetreffende serveromgeving.
- Indien toepassing op de presentatielaag wordt voorzien dient te worden nagegaan of de implementatie van RESTful Services noodzakelijk is.
- Ook dient te worden nagegaan of een alternatief toepassingsscenario, waarbij de service via Internet beschikbaar wordt gesteld, economische of technische voordelen biedt, en of de serviceaanbieder een dergelijk scenario ondersteunt.

Ten tweede, met welke aspecten moet rekening worden gehouden wanneer functies voor het valideren, actualiseren en verrijken van gegevens voor SOA geschikt worden gemaakt?

- De toepassingsscenario's moeten duidelijk worden beschreven. Erg belangrijk daarbij is de vraag of de services worden ingezet in de context van de business logic of van de presentatielogica. In het eerste geval vindt de integratie plaats op de applicatieserver, in de Enterprise Service Bus of op een BPEL Engine, in het tweede geval in een grafische interface, die weer eigen eisen kan stellen. Afhankelijk daarvan volgt de beslissing of op SOAP gebaseerde services en/of RESTful Services meer geschikt zijn.
- De doelsystemen en -talen, waarop/waarin de service zal worden gebruikt, moeten worden vastgelegd. Daarvan hangt de beslissing af van hoe complex de modellering wordt met betrekking tot XML-structuren en XML-datatypen van de aanroepresultaten.
- De service moet 'stateless' worden ontworpen, dat wil zeggen: de service moet functioneren zonder dat tussen twee opeenvolgende oproepen een interne toestand wordt opgeslagen. Mocht dit laatste wel nodig zijn, dan moet de toestand bij de volgende aanroep doorgegeven of op geschikte wijze persistent gemaakt worden.
- Schaalbaarheid van de service moet zijn gewaarborgd: als de webservice is aangewezen op global resources, bijvoorbeeld een databaseverbinding, dienen deze via een resource pool in de servercontainer te worden beheerd. Anders worden deze resources snel een belemmering voor de schaalbaarheid.
- De service moet geschikt zijn voor het Internet, dat wil zeggen: het mag voor het functioneren van de service niet uitmaken of deze via een lokaal netwerk of via Internet wordt aangeboden. Daarmee nemen de toepassingsmogelijkheden enorm toe.

Walter Eichhorn (walter.eichhorn@uniserv.com) is Product Manager bij Uniserv GmbH.