Het bouwen van een repository voor een Business Rules Engine

Rules Engine wordt vereiste component

Malcolm Chisholm

Business rules bestaan voor het merendeel uit metadata. Ze representeren logica die bijna altijd bedoeld is om data te manipuleren. 'Data' verschijnen in business rules als metadata, zoals attribuut- of kolomdefinities, soms ook datawaarden. Als business rules grotendeels bestaan uit metadata, dan is alles waarin ze worden opgeslagen per definitie een repository. Een repository is tenslotte gewoon een database waarin metadata worden opgeslagen.

Er zijn twee typen repository's waarin business rules worden opgeslagen:

- Repository's alleen bedoeld voor de opslag van business rule-definities; ze worden vooral aangewend door eindgebruikers;
- Repository's die niet alleen de rule definities opslaan, maar ook zorgdragen voor de uitvoering als onderdeel van business rule engines.

In dit artikel bekijken we de belangrijkste componenten van de repository-structuur die nodig is om een business rules engine te ondersteunen. Een flink deel hiervan heeft ook betrekking op repository's die business rule-definities herbergen. Er zijn echter verschillen tussen de twee typen repository's; een repository voor rule-definities is niet gewoon een subset van een repository die rules engines ondersteunt. In tegenstelling tot de eerste, bevat de laatste bijvoorbeeld veel metadata die het analystische proces ondersteunen.

In dit artikel refereert Malcolm Chisholm aan op de markt verkrijgbare Business Rules Engines.
Enkele leveranciers op dit gebied zijn CA, FairIsaac, ILOG, Pegasystems, RuleBurst en YASU Technologies.

Meer informatie over Business Rules kunt u vinden op de website van het Business Rules Platform Nederland, www.brplatform.org. Uitgebreide artikelen over Business Rules vindt u in het septembernummer van zusterblad Business Process Magazine.

Functionaliteit

Een vraag die veel gesteld wordt, is waarom een onderneming *überhaupt* zelf een business rules engine zou bouwen. Het is tenslotte een omvangrijke klus en er zijn op de markt veel generieke rules engines verkrijgbaar. Dat is waar, maar in de praktijk komt het steeds meer voor dat business rule-functionaliteit aan applicaties wordt toegevoegd om ze additionele flexibiliteit te geven. Dergelijke functionaliteit representeert een paar procent van de algehele functionaliteit van de applicatie. Een bedrijf kan bijvoorbeeld een sales-systeem hebben, waarin rules engine functionaliteit is ondergebracht om de kortingen te berekenen die aan klanten op basis van performance in het verleden wordt gegeven. In dat geval zal de applicatie veel 'fixed' logica bevatten, die eenmalig door de programmeurs wordt geïmplementeerd en waarvan verwacht wordt dat er daarna nauwelijks verandering in zal komen. De applicatie zelf heeft ook logica nodig, die bij elke transactie verandert. Deze logica kan niet tevoren worden voorspeld en moet in een short-time frame worden geïmplementeerd. Het is niet praktisch om programmeurs de implementaties te laten doen, dus het enige alternatief is om een vorm van rules engine-functionaliteit te creëren.

Database design metadata

Alle rules engine-functionaliteit manipuleert data. Daarom moet elke repository bedoeld voor een rules engine, metadata bevatten die data representeren. Deze kunnen meestal uit de datamodellen gehaald worden; veel datamodelleer-tools bieden de mogelijkheid om er metadata uit te halen. Rules engines hebben die metadata nodig voor zowel het logische datamodel (bijvoorbeeld entiteits- en atribuutdefinities) als voor de fysiek geïmplementeerde database (zoals tabel- en kolomnamen, kolom datatypes).
Dit is een probleem voor bedrijven die hun datamodellering niet van tevoren doen. Zulke bedrijven hebben meestal logische data-modellen die business definities bevatten, maar die niet corresponderen met de fysiek geïmplementeerde databases. Omgekeerd levert reverse engineering van een fysiek geïmplementeerde database naar een datamodel niet de benodigde business-definities die nodig zijn voor de rules engine.

Een rules engine heeft ook informatie nodig over verwantschappen. Het kan bijvoorbeeld nodig zijn een kolom met bedragen op te tellen in een *parent table*-kolom via een business rule.

Dergelijke 'aggregatie' rules zijn alleen toegestaan tussen *parent*- en *child*-tabellen. Als de informatie over de verwantschap niet in de rules engine repository is opgeslagen, kunnen dit soort rules niet worden gedefinieerd en betrouwbaar geïmplementeerd. Bovendien kunnen metadata over een eenvoudige parent/child-verwantschap niet voldoende zijn. Business rules hebben vaak te maken met verwantschappen tussen een tabel en haar grootouder- of overgrootouder-tabellen. In de praktijk betekent dit dat alle mogelijke navigatiepaden in een database moeten worden gedefinieerd in de rules engine repository. Nogmaals, het moet mogelijk zijn dergelijke informatie uit de datamodellen te verkrijgen. In die gevallen biedt reverse engineering van de fysieke database naar een datamodel weinig soelaas, omdat de verwantschappen niet kunnen worden afgeleid tijdens het proces, tenzij de foreign key-relaties zijn gedefinieerd in de databasestructuur – wat meestal niet het geval is. Dit is een veelvoorkomend probleem bij het bouwen van rules engine-functionaliteit, dat de noodzaak voor goede forward engineering van databases illustreert.

Reference datawaarden

Een ander belangrijk probleem bij het definiëren van business rules, is dat actuele reference data-waarden nodig zijn. Reference data zijn ook bekend als codekolom- of domeinwaarden. Het zijn de kleine tabellen die meestal alleen maar een codekolom en een beschrijvende kolom bevatten, en meestal maar een paar rijen. Om een paar voorbeelden te noemen: Land, Type Klant, Valuta, Productcategorie. Veel IT-professionals die zich bezighouden met metadata management houden er niet van om met waarden uit de fysieke database-tabellen om te gaan. Ze zien dat niet als metadata en negeren het dus. Business rules hebben echter reference data nodig. Vele business rules bevatten directe verwijzingen naar actuele datawaarden en dat zijn altijd reference data. Producttabellen bijvoorbeeld hebben meestal vele ermee verbonden reference data-tabellen, die dienen om de records erin te categoriseren. Records in Productcategorie A zouden een hele verzameling business rules kunnen hebben die alleen daarop toegepast worden, terwijl records in categorie B gebruik maken van een volkomen andere verzameling business rules.

Business rules die reference data nodig hebben, gebruiken bijna altijd codewaarden. Codewaarden zijn echter lastig om mee te werken; je hebt codebeschrijvingen nodig. Maar zelfs beschrijvingen kunnen tekort schieten. Zo kan in de business rules bijvoorbeeld de landcode 'PRC' voorkomen, met als omschrijving 'People's Republic of China'. Voor een bepaalde onderneming vallen Hong Kong en Taiwan daar echter niet onder, maar dat blijkt niet uit de beschrijving. Om business rules te definiëren die reference data gebruiken, zijn dus ook definities van de reference data nodig.

Reference data worden bijna altijd als een ondergeschoven kindje behandeld, zelfs in de op de markt verkrijgbare business rules engines. Hoe dan ook, elke repository voor een business rules engine moet reference data ondersteunen. Dat betekent dat als de reference data-waarden in een database zijn gedefinieerd, ze ook bekend gemaakt moeten worden aan de business rules engine repository. Dit is de grote uitdaging voor de ontwerper van de business rules engine repository. Maar al te vaak wordt het als extreem moeilijk beschouwd en kunnen de ontwerpers de uitdaging niet aan. Ze nemen hun toevlucht tot oplossingen als het direct in de rules engine intypen van de vereiste codes. Dit degradeert codewaarden tot tekst en maakt het onmogelijk om de additionele metadatadiensten te bieden die in toenemende mate van business rules repository's worden verwacht. Het verkrijgen van bijvoorbeeld een lijst van business rules waarin een bepaald item van reference data wordt gebruikt, is veel lastiger en minder betrouwbaar als de codewaarden simpelweg in de business rules zijn getypt.

Inflatie van attributen en kolommen

Omdat business rules data manipuleren, is het belangrijk dat de uitkomsten van business rules in een database persistent zijn. Als dat niet zo is, wordt het *auditen* van business rule-toepassingen onmogelijk; de rules moeten herhaaldelijk worden afgevuurd om hun gevolgen te zien op andere rules. De beste manier om deze problemen op te lossen, is ervoor te zorgen dat elke rule een target-kolom een update geeft. Goed datamanagement vereist dat als een business rule een kolom een update geeft, dat alleen kan gebeuren door één rule en niet twee, of nog meer. Dat heeft consequenties. Het is erg verrassend, misschien zelfs schokkend, om te zien hoe business rule projecten kunnen leiden tot een aanmerkelijke toename van het aantal attributen in een datamodel of kolommen in een fysiek geïmplementeerde database.

> Veel vakmensen op het gebied van business rules éisen dat rule-afhankelijkheden automatisch worden vastgesteld

Terugkijkend is het eenvoudig te begrijpen hoe dat kan gebeuren. Stel, we hebben een bankapplicatie met een attribuut 'Account Balance', met als definities iets als 'Geldbedrag op dit moment beschikbaar voor de klant.' Deze kolom kent echter verschillende berekeningen voor particulieren, bedrijven of medewerkers. Dat betekent dat we drie verschillende business rules nodig hebben om 'Account Balance' te kunnen berekenen – wat een vergissing is, want een kolom mag maar één definitie hebben en dus maar één business rule om het te updaten. Elk van de drie business rules moet zijn eigen target-kolom hebben, dus moeten we een 'Corporate Account Balance', een 'Individual Account Balance' en een 'Employee Account Balance' toevoegen. We zouden 'Account Balance' daarna kunnen verwijderen, maar misschien is het wel

handig om hem te hernoemen als 'Consolidated Account Balance' en een business rule te creëren die het de som maakt van de andere drie. Business rules representeren een bepaald niveau van precisie en detail dat veel hoger is dan de tekstuele attribuut-definities zal die zich in datamodellen bevinden. Business rules-projecten zullen dus onvermijdelijk leiden tot definitie van de vereiste additionele attributen en kolommen.
Het is erg moeilijk om deze consequentie van het werken met business rules in te schatten, tot men een keer een project heeft gedaan. Daar komt bij dat het ook eisen stelt aan het ontwerp van de repository. Het moet mogelijk zijn om op een eenvoudige manier kolommen toe te voegen aan de databasestructuur, waar de business rules repository op opereert. Dat betekent dat de mogelijkheid aanwezig moet zijn om de metadata voor deze kolommen aan de business rules repository toe te voegen; de repository moet dan de database-kolommen waarop hij opereert kunnen aanpassen om nieuwe kolommen te kunnen aanmaken. Helaas betekent dit ook dat het datamodel waarmee aan het

> Een belangrijk probleem bij het definiëren van business rules is dat actuele reference data-waarden nodig zijn

business rule engine project werd begonnen, snel veroudert omdat de rules engine zelf nieuwe kolommen toevoegt. Dit kan tot grote organisatorische problemen leiden. Het team dat verantwoordelijk is voor het datamodel moet van elke verandering op de hoogte worden gesteld. Misschien kunnen ze toegang krijgen tot de business rules repository waar ze kunnen bepalen welke de nieuwe kolommen zijn. Een andere mogelijkheid is, dat er meta-data uit de business rules repository kunnen worden onttrokken die weer worden geïmporteerd in het datamodellering-tool om het datamodel te herbouwen. Ik heb met beide methoden gewerkt en geef uiteindelijk de voorkeur aan de laatste. Er is nog een mogelijkheid, namelijk het simpelweg gebruiken van de business rules repository in plaats van het datamodel. In ondernemingen met een beperkt budget, zou dit wel eens de meest praktische aanpak kunnen zijn. De business rules repository heeft dan wel extra datamodelleringfunctionaliteit nodig. Dat is een lastig probleem, maar het is beter het te constateren, voordat wordt begonnen met het bouwen van business rules engine-functionaliteit.

Business-processen

Het bouwen van een repository waarin business rules kunnen worden gedefinieerd is één aspect. Het afvuren van de business rules vereist additionele metadata, die ook weer beheerd moet worden. Sommige IT-professionals denken aan één pakket business rules per onderneming en vervolgens kunnen de onderlinge afhankelijkheden worden uitgewerkt, zodat ze in de juiste volgorde kunnen worden afgevuurd. Dit is helaas niet waar.
Een rule voor, zeg, het berekenen van de voordelen van goede gezondheid bij een medewerker heeft geen enkele relatie met een rule voor het becijferen van kortingen voor een distributeur.
Gelukkig zijn bedrijven veel meer geïnteresseerd in het creëren van een rules engine met beperkte functionaliteit voor uiterst specialistische en goed gedefinieerde applicatie-componenten. Dit betekent dat er rules-pakketten worden gedefinieerd die wel een relatie met elkaar hebben, maar het kan nog steeds zo zijn dat het niet altijd mogelijk is om de volgorde waarin de rules worden afgevuurd automatisch vast te stellen.
Dit probleem kan worden opgelost door de business-processen in de repository vast te leggen. Elk business-proces correspondeert met een pakket rules. Binnen elk proces kunnen de afhankelijk-heden van de rules automatisch worden bepaald, zodat de rules in de juiste volgorde kunnen worden afgevuurd.
Het vastleggen van business-processen en er vervolgens rules aan toewijzen, is een ontwerp dat erg veel flexibiliteit schept. Het betekent dat een rule kan worden hergebruikt in verschillende processen. Dat is bijzonder belangrijk, bijvoorbeeld voor rules die banksaldo's herberekenen. Dergelijke regels moeten telkens als de financiële informatie over een bankrekening wijzigt gedraaid worden, ongeacht het business-proces. Een alternatief is om de business rules strikt binnen de business-processen vast te leggen, dat liever dan onafhankelijk ervan. Dit lijkt misschien logischer in termen van programmering, maar het belemmert echt het hergebruik van business rules.

De benadering waarbij rule-afhankelijkheden automatisch worden uitgewerkt, heeft ook zijn beperkingen. Dat is echter geen populaire zienswijze. Veel vakmensen op het gebied van business rules éisen dat rule-afhankelijkheden automatisch worden vast-gesteld. Als mensen de volgorde van uitvoering van business rules bepalen, dan wordt dit gezien als 'procedureel' en een incorrecte aanpak van het ontwerp. In werkelijkheid kunnen er hele goede redenen zijn om de volgorde waarin de rules worden afgevuurd te specificeren. Financiële 'watervallen' zijn een goed voorbeeld. Als ik mijn rekeningen aan het eind van de maand betaal, maak ik eerst mijn hypotheekrente over, dan energie, water en telefoon. Vervolgens betaal ik mijn credit card rekeningen op volgorde van belangrijkheid. Zo'n 'betalingswaterval' komt heel veel voor bij complexe financiële transacties; de volgorde waarin de betalingen plaatsvinden is zelf een soort business rule die moet worden vastgelegd en die niet automatisch kan worden bepaald door welk type repository-functionaliteit we ook bouwen. Een rules engine repository moet de metadata kunnen ondersteunen, nodig voor de vaststelling van de volgorde waarin de rules worden uitgevoerd.

Audit van Rule Execution

Een ander aspect aan het ontwerp van de repository voor een business rules engine, is de noodzaak om de uitvoering van business rules te volgen. Ter herinnering: de adoptie van rules

engine-functionaliteit wordt onder meer gedreven door de vraag naar logica die snel kan worden geïmplementeerd door de eindgebruikers, zonder tussenkomst van programmeurs. Dit lijkt misschien wenselijk en heeft vele voordelen, maar heeft ook een erg groot nadeel. Dat heeft te maken met de mate waarin programmeurs testen draaien en bugs herstellen. In feite bestaat ongeveer de helft van de tijd en moeite die gestoken wordt in de traditionele systeemontwikkelings life cycle, uit testen en debugging.

> De business rules repository heeft wel extra datamodellering-functionaliteit nodig

Als eindgebruikers zelf nieuwe rules gaan definiëren in een productie-applicatie is de hulp van programmeurs niet beschikbaar. Als een gebruiker een bug constateert is hijzelf verantwoordelijk voor het herstel ervan. Ook dit is een consequentie van de business rules benadering, die pas wordt begrepen als men er in de praktijk mee wordt geconfronteerd.

Om de gebruikers toch behulpzaam te zijn bij het debuggen, zou de business rules repository zo moeten zijn ontworpen, dat de uitvoering van rules kan worden gevolgd. Elke rule die wordt afgevuurd moet worden vastgelegd, net als de output van de rule. Deze informatie moet voor de gebruikers beschikbaar zijn op een voor hun begrijpelijke manier. Het is mijn ervaring dat dergelijke audit-functionaliteit de prestaties van de rules engine onvermijdelijk vertraagt. Daarom zou het optioneel moeten zijn, zodat het alleen gedraaid wordt als dat nodig is. Ik heb ook ondervonden dat het tamelijk lastig is om audit-rapportages te leveren die zinvol zijn voor gebruikers. De neiging bestaat om deze rapporten er uit te laten zien als ondoorgrondelijke logs. Met wat moeite kunnen ze echter begrijpelijk worden gemaakt.

Tot slot

Er zitten nog veel meer aspecten aan het ontwerp van een repository die een business rules engine ondersteunt, maar de belangrijkste zijn hiervoor behandeld. Het is erg belangrijk om te onthouden dat het niet het doel moet zijn om een generieke rules engine te bouwen. Het is verstandiger om beperkte rules engine-functionaliteit te implementeren, om zo de nodige flexibiliteit aan traditionele applicaties te geven. In de toekomst zal de toenemende complexiteit van informatiesystemen er toe leiden dat rules engine-functionaliteit een gebruikelijke, zelfs vereiste component van die systemen zal zijn.

Noot

In geval van discussies geeft de originele Engelstalige tekst van dit artikel de doorslag. Deze tekst is te vinden op onze website www.dbm.nl, in het hoofdmenu onder Specials/Extra materiaal.

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

Update

Omzetgroei Oracle EMEA

Met een groei van 83 procent in omzet uit applicaties in de EMEA-regio in vergelijking met dezelfde periode vorig jaar, groeit de applicatietak van Oracle harder dan de concurrentie. Klanten reageren positief op Oracle's strategie om klanten de keuze te geven hun bestaande systemen te blijven gebruiken totdat de tijd rijp is om nieuwe technologieën te gaan gebruiken.
Oracle Applications Unlimited programma biedt zekerheid dat de support en productontwikkeling voor Oracle, Siebel, PeopleSoft, and JD Edwards applicaties voor de toekomst gegarandeerd is. Veel organisaties in EMEA gingen over tot de implementatie of uitbreiding van de Oracle Database, Oracle Fusion Middleware, Oracle Grid technologie en Oracle Applicaties waaronder de Oracle E-Business Suite, PeopleSoft Enterprise, Siebel CRM, JD Edwards EnterpriseOne and JD Edwards World in het eerste kwartaal van boekjaar 2007.
Nederlandse klanten die vorig kwartaal kozen voor Oracle zijn onder andere: CWI, ING Bank, Rabobank, Meester Stegeman-Sara Lee Foods, Coöperatie VGZ-IZA en ROC Nijmegen.
Zie www.oracle.com

SAS: toenemende vraag naar voorspellende analyses

SAS noteert een sterke groei in de verkoop van SAS Forecast Server. SAS heeft een leidende marktpositie met oplossingen voor marktanalyses met voorspellende waarde. Sinds de introductie eind 2005, heeft SAS meer dan 6500 licenties aan 4000 klanten wereldwijd verstrekt en de verkoop groeit. Dit is te danken aan het feit dat steeds meer bedrijven de waarde van voorspellende analyses inzien en de accuratesse van de SAS-oplossingen waarderen. SAS Forecast Server is waardevol voor bedrijven in bijvoorbeeld de financiële dienstverlening, leveranciers van consumentenartikelen, productiebedrijven, voor de detailhandel, en veelsoortige dienstverleners als telecommunicatie- en nutsbedrijven. De functies van SAS Forecast Server, waarop octrooi is aangevraagd, levert automatisch voorspellingen door patronen en trends door de loop van de tijd te ontdekken. De interactieve gebruikersinterface maakt dat zelfs onervaren gebruikers de geavanceerde technologie eenvoudig kunnen gebruiken.
SAS Forecast Server is gebaseerd op het Enterprise Intelligence Platform.
Zie www.sas.com