ESPECIFICAÇÃO DE REGRAS DE NEGÓCIO PARA BANCO DE DADOS RELACIONAL-OBJETO

Franz Novillo Torrico (fnovillo@usa.net)
Astério Kiyoshi Tanaka (tanaka@uniriotec.br)[1]
Ana Maria de Carvalho Moura (anamoura@ime.eb.br)

Instituto Militar de Engenharia – IME
Departamento de Engenharia de Sistemas
Praça General Tibúrcio 80, Praia Vermelha – CEP: 22290-270 – Rio de Janeiro, RJ – Brasil

ABSTRACT

Object-Relational Database Management Systems (DBMSs) are a natural evolution of relational DBMSs, incorporating some object-oriented characteristics and support to active rules. DBMS active functionalities make it possible to create application development environments with the ECA (event-condition-action) paradigm implementing business rules. In this work, we describe a Metamodel for Business Rules, which is conceived according to the four-layer architecture (meta-metamodel, metamodel, model and data), and results in the Extended $(ER)^2$ Model (Entity-Relationship-Event-Rule Model with Object-Oriented features). This allows capturing of internal object behavior through the concept of state, and active object behavior in the form of events and rules. These concepts have been incorporated into a database design tool prototype with mapping to object-relational DBMS. In this tool, data are modeled using the Unified Modeling Language (UML) and business rules are modeled with the extended $(ER)^2$ Model.

Keywords: Databases, Business Rules, Database Modeling and Design.

1. Introdução

Os Sistemas de Gerência de Bancos de Dados Relacionais-Objeto (SGBDRO), às vezes chamados de "universais", surgiram comercialmente em 1997, como uma evolução natural dos SGBDs relacionais. Os SGBDRO acrescentam algumas características da orientação a objetos aos SGBDs relacionais, além de um subsistema de suporte a regras ativas, já presentes em SGBDs relacionais mais avançados.

SGBDRO possuem capacidade ativa através de regras ou "triggers", podendo ser classificados como bancos de dados ativos. Bancos de dados ativos são sistemas com toda a funcionalidade dos bancos de dados convencionais acrescida da capacidade de detectar a ocorrência de eventos, monitorar condições específicas sobre o estado do banco de dados e executar algumas ações independentemente de qualquer solicitação externa ao sistema [15].

Pela inexistência de uma representação conceitual de regras e eventos, que permita especificar regras de negócio ativas e traduzi-las nas construções existentes nos SGBDRO, apresentamos neste

[1] Professor do Departamento de Informática Aplicada da UNI-RIO (Universidade de Rio de Janeiro) e Pesquisador Colaborador do Departamento de Engenharia de Sistemas do IME, parcialmente apoiado pelos projetos CNPq processo nº 350652/94-5 e PROTEM-CC/INRIA nº 68.0139/98.2.

artigo a estrutura e semântica de um metamodelo para a especificação de regras de negócio na modelagem conceitual dos sistemas de informação, com abordagem orientada a objetos.

O modelo $(ER)^2$ estendido é um complemento ao modelo de classes e objetos da UML com os conceitos de eventos e regras especificados no modelo $(ER)^2$ [14]. Esses conceitos são mapeados para as construções existentes em SGBDRO.

O artigo está organizado da seguinte forma. A seção 2 descreve brevemente os conceitos básicos das regras de negócio. A seção 3 apresenta uma análise e comparação de metamodelos e ferramentas de modelagem conceitual com regras de negócio. A seção 4 descreve a estrutura do metamodelo e o modelo $(ER)^2$ estendido. Finalmente, a seção 5 apresenta as conclusões, com as perspectivas de trabalhos futuros.

2. Regras de Negócio

Os sistemas de informação abrangem, com freqüência, um grande volume de conhecimento organizacional formalizado. Tal conhecimento é especificado como *regras de negócio* [2], já antes definidas por Bell [3] como sendo “declarações que apresentam a maneira de como o negócio está sendo feito, além das diretrizes e restrições com respeito a estados e processos em uma organização”. Surge com isso a necessidade de se modelar regras de negócio, enfatizando a sua importância em um esquema conceitual, que, segundo Loucopoulos [12], representa todas as regras que governam o universo de discussão, devendo estar definidas dentro de um esquema conceitual. A partir desta necessidade, tais regras de negócio têm sido amplamente utilizadas pelas organizações como forma de melhorar a qualidade de seus sistemas.

Atualmente, os SGBDs têm a capacidade de executar ações através do uso de regras ativas baseadas no paradigma Evento-Condição-Ação (ECA)[15]. O componente "Evento" indica o momento do disparo da regra, a "Condição" indica um predicado a ser avaliado e a "Ação" o que tem de ser feito, caso a condição seja válida. Estes mecanismos ativos, denominados gatilhos (*triggers*) em bancos de dados, proporcionam uma funcionalidade aos SGBDs, cuja principal aplicação tem sido a manutenção de integridade e restrições de acesso aos bancos de dados. Desde que devidamente especificadas no esquema conceitual, regras de negócio podem ser implementadas com o paradigma ECA, ampliando a sua aplicação a problemas do mundo real.

2.1 Componentes das Regras Ativas

Para a especificação de regras ativas e de modo a apresentar a importância prática do paradigma ECA, os componentes evento, condição e ação merecem um interesse especial por sua riqueza semântica e sua complexidade detalhadas a seguir:

- Componente Evento

O evento é o componente disparador da regra. Álgebras de eventos, como, por exemplo, as usadas nos sistemas *Snoop* (linguagem de especificação para SGBDs ativos) *e SAMOS (Swiss Active Mechanism Based Object-Oriented Database System)* foram propostas para formalizar o conceito de evento.

- A álgebra de evento Snoop é uma linguagem de especificação para SGBDs ativos desenvolvida na Universidade da Flórida, EUA [5]. Um evento em Snoop é definido como uma ocorrência atômica de uma operação no banco de dados, uma ação externa explícita ou um fato temporal (Figura 1).

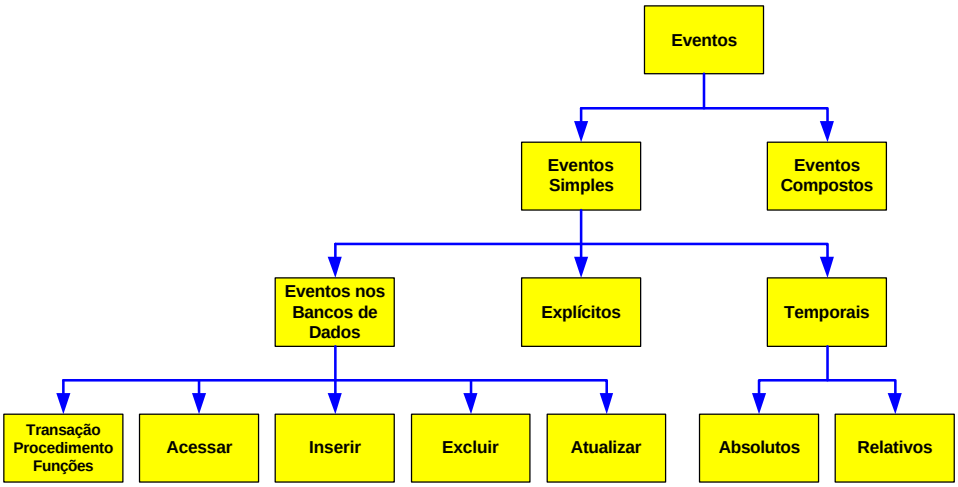

Figura 1: Classificação de Eventos Snoop

Eventos podem ser compostos através de operadores lógicos como conjunção, disjunção ou seqüência, resultando em ocorrências atômicas.

- A álgebra de eventos SAMOS [8] foi desenvolvida na Universidade de Zurich, Suíça, utilizando como base o SGBD Orientado a Objeto (SGBDOO) ObjectStore. Semelhantemente ao Snoop, os eventos propostos por SAMOS são simples ou compostos, com pequenas diferenças quanto à classificação dos eventos simples e quanto aos operadores de eventos compostos.

– Componente Condição

Condições das regras podem ser classificadas de acordo com os tipos de restrições de integridade . À semelhança do evento, o componente condição pode ser simples e composto [9].

- *Condições simples* são predicados atômicos aplicados sobre o estado corrente do banco de dados, podendo incluir operadores booleanos unários tal como NOT.
- *Condições compostas* resultam da aplicação dos operadores booleanos binários AND ou OR em condições simples ou compostas.

– Componente Ação

Para o componente ação das regras ativas, foram propostas várias classificações, dependendo da natureza das operações que executam (operações de banco de dado ou das aplicações). De forma semelhante aos componentes evento e condição, as ações também podem ser classificadas em ações simples e compostas [10].

- *Ação simples* consiste em exatamente uma operação que não seja refinada posteriormente.
- *Ação composta* é uma sucessão de algumas operações a serem executadas.

2.2 Classificação das Regras de Negócio

De forma genérica, regras de negócio, sob forma de regras ativas, podem ser classificadas em dois tipos [11]:

– *Regras de Integridade*: representam a semântica das restrições de integridade, que são estendidas tornando explícito o disparo de eventos e a reação sobre uma violação de restrição. Exemplo:

ON (estoque de produto atualizado)

IF (novo estoque < 0)
THEN emitir mensagem de erro
"O estoque não pode ser menor que zero"

– *Regras de Automatização*: descrevem a lógica de execução de uma tarefa; assim, a violação da condição não tem a semântica de um erro. Exemplo:

ON (estoque de produto atualizado)
IF (novo estoque < limite para pedido de reposição)
THEN fazer um pedido do produto

3. Metamodelos e Ferramentas de Modelagem de Regras de Negócio

À guisa de revisão de trabalhos correlatos, é feita a seguir a análise e comparação de metamodelos reportados na literatura bem como ferramentas de modelagem conceitual com especificação de regras de negócio: o *BROCOM* (Business Rule Oriented Conceptual Modeling), desenvolvido na Universidade de Bern, Suíça [9]; o *RIM* - Rechnergestütztes InformationsManagement (gerência de informação baseado em computador), desenvolvido pelo Instituto de Sistemas de Informação, da Universidade de St. Gall, Alemanha [7]; e o *(ER)*^{2}, proposto em [14] e desenvolvido no Instituto Militar de Engenharia, Rio de Janeiro, Brasil [1,6].

A Tabela 1 apresenta uma análise comparativa desses metamodelos, a partir dos seguintes critérios: os componentes do metamodelo como objetos do universo de discussão e os fatos organizacionais no metamodelo que abrange tipos de metaclasses que são relevantes e estão relacionadas com as regras de negócio.

Os metamodelos apresentados contemplam apenas modelos tipicamente relacionais. Nestes modelos existe efetivamente uma carência de componentes que permitam projetar regras de negócio incluindo a parte dinâmica com o controle interno do objeto, e em particular para SGBDRO, como pode ser observado na tabela 1.

TABELA 1: Comparação dos metamodelos

Metamodelo	Modelo de dados	Componentes ativos			Componente de processo	Controle interno do objeto
		Evento	*Condição*	*Ação*		
BROCOM	Entidade Relacionamento	Simples e Compostos	Condição	Ação	Especificado por Regra de Negócio	Não
RIM	Entidade Relacionamento	Eventos de Negócio associados a Funções de Negócio	Restrições de Integridade implícitas	Função de Negócio	Especificado por Função de Negócio	Não
$(ER)^2$	Entidade Relacionamento	Evento de banco de dados	Condição e Ação são partes do componente Regra		Ausente	Não

A partir desta comparação verificou-se a necessidade de se criar um metamodelo que permita especificar regras de negócios no contexto da modelagem orientada a objetos.

3.1 Ferramentas de Modelagem de Regras de Negócio.

A Figura 3 abaixo ilustra as ferramentas de modelagem associadas aos metamodelos descritos.

– Modelo $(ER)^2$

O modelo $(ER)^2$ (Entidade-Relacionamento-Evento-Regra) [14] incorpora ao Modelo Entidade-Relacionamento (MER) a especificação de regras e eventos, como objetos de primeira classe, para a modelagem de banco de dados ativos. As regras e os eventos passam a fazer parte do modelo, com a notação gráfica e identificação própria. Em um diagrama $(ER)^2$, um evento é representado por um círculo e uma regra por um paralelogramo. As conexões entre eventos e regras são intuitivas.

– Modelo Evento-Condição-Ação-se-Ação-senão (ECA^2)

Redes de Petri [9] são modelos formais para especificar o fluxo de controle em um sistema. A rede ECA^2, resultante da aplicação desses símbolos, é também uma Rede de Petri de alto nível que pretende representar exatamente o comportamento de um sistema e pode ser empregado como base para a simulação dos processos.

– Modelo de Processamento Conceitual

A metodologia Merise: An Information System Design and Development Methodology, in: Information & Management [9] emprega o Modelo de Processamento Conceitual (CPM – Conceptual Processing Model) para a especificação de processos. Este modelo abrange eventos de ativação, operações (sincronizadas), e eventos resultantes. Uma operação consiste em uma ou mais tarefas que estão baseadas no gerenciamento de regras e são executadas seqüencialmente. Em CPM, um evento é visto como um portador de propriedades, que correspondem aos atributos no modelo de dados do Merise.

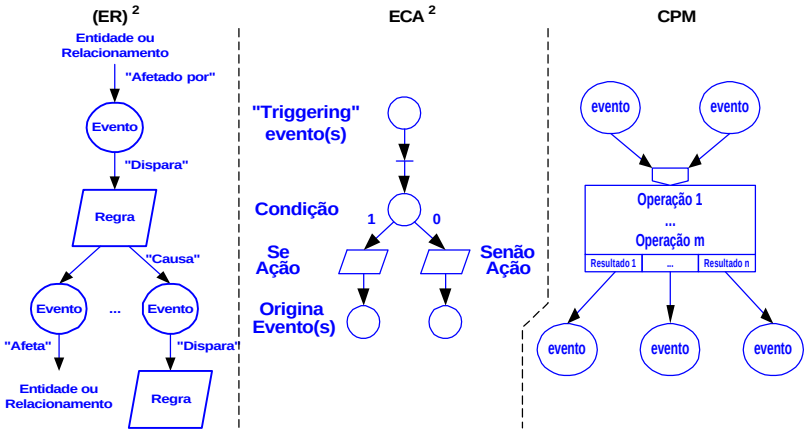

Figura 3: Ferramentas de Modelagem Conceitual

Desta comparação, chegou-se a conclusão que uma rede ECA^2 simplificada tem características similares de representação gráfica e semântica ao modelo $(ER)^2$ e que CPM apresenta problemas como na lógica de eventos compostos que não pode ser representada, só podendo ser descrita literalmente. Daí a justificativa da escolha do modelo $(ER)^2$ como base deste trabalho.

4. Metamodelo para Regras de Negócio em SGBDRO

Neste artigo, o metamodelo proposto para a modelagem conceitual foi concebido com base no modelo de classes da UML [4] e tem como premissa o estudo realizado na seção anterior sobre metamodelos, especialmente aqueles que têm um enfoque das regras de negócio estruturadas para o modelo $(ER)^2$.

A seguir descreve-se o metamodelo para regras de negócio, concebido segundo a arquitetura de quatro camadas de forma de atender a interoperabilidade e escalabilidade [13]. Conforme ilustrado na Figura 4, o metamodelo está composto das metaclasses descritas a seguir.

4.1 Componentes do modelo de dados

A MetaClasse "Componentes do Modelo de Dados" é uma entidade genérica formada por outras MetaClasses, também consideradas como objetos de primeira classe, neste modelo, a saber: “Regra de Negocio”, “Classe” e “Associação”.

Além disso possui uma MetaAssociação com a MetaClasse “Esquema de dados”, determinando que os componentes do modelo de dados estão *incluídos* em um conjunto finito do Esquema de Dados. Os componentes do modelo de dados são descritos a seguir.

4.1.1 Regra de Negócio

A MetaClasse Regra de Negócio, estruturada como uma regra $(ER)^2$, é um agregado constituído por dois componentes básicos: evento e regra. Está associada, ainda, a outros três tipos de MetaClasses: “Classe”, “Estado” e “Processo”.

A semântica das associações relacionadas a essa MetaClasse é descrita a seguir:

1. Cada regra de negócio pode estar *refinada* por um processo específico.
2. Regras de negócio podem ser *especificadas* por processos específicos.
3. Regras de negócio podem *afetar* uma ou mais classes.
4. Regras de negócio podem *mudar* um determinado estado.

Uma regra de negócio pode ser habilitada ou desabilitada num modelo de dados através do atributo “status” especificado na metaclasse.

Se o evento de uma regra de negócio ativa mais de uma regra, só uma será disparada, devido ao atributo “prioridade” de disparo especificada na metaclasse.

4.1.1.1 Evento

A MetaClasse “Evento”, é parte da MetaClasse “Regra de Negócio”. Essa é uma ocorrência significativa, que tem localização no tempo e espaço, e deve ser reconhecida e reagida pelo sistema em estudo.

A MetaClasse Evento é relacionada através tipos de MetaAssociação entre MetaClasses “Evento” (recursivamente), “Regra”, “Ação” e “Agente Externo”, segundo a descrição a seguir:

1. Cada evento pode *disparar* um conjunto finito de regras;
2. Eventos podem ter *origem* em no máximo uma ação;
3. Eventos podem ter *origem* em Agentes Externos;
4. Um evento pode ser ativado *temporalmente* pela ocorrência de pontos predeterminados no tempo.
5. Um evento pode ser composto pela *composição* de eventos simples

O Metamodelo considera a MetaClasse evento como uma entidade genérica que abrange as seguintes MetaClasses: “Evento de Banco de Dados”, “Evento Externo” e “Evento Temporal”.

Por sua vez a MetaClasse *Evento de Banco de dados* inclui quatro operações para manipulação de dados. São elas: acessar (“*access*”), inserir (“*insert* ”), excluir (“*delete*”) e atualizar (“*update*”). Eventos ocorrem no começo ou ao final das transações, incluindo a chamada a procedimentos, funções e/ou métodos de aplicação acionados no começo ou ao final da execução de um método.

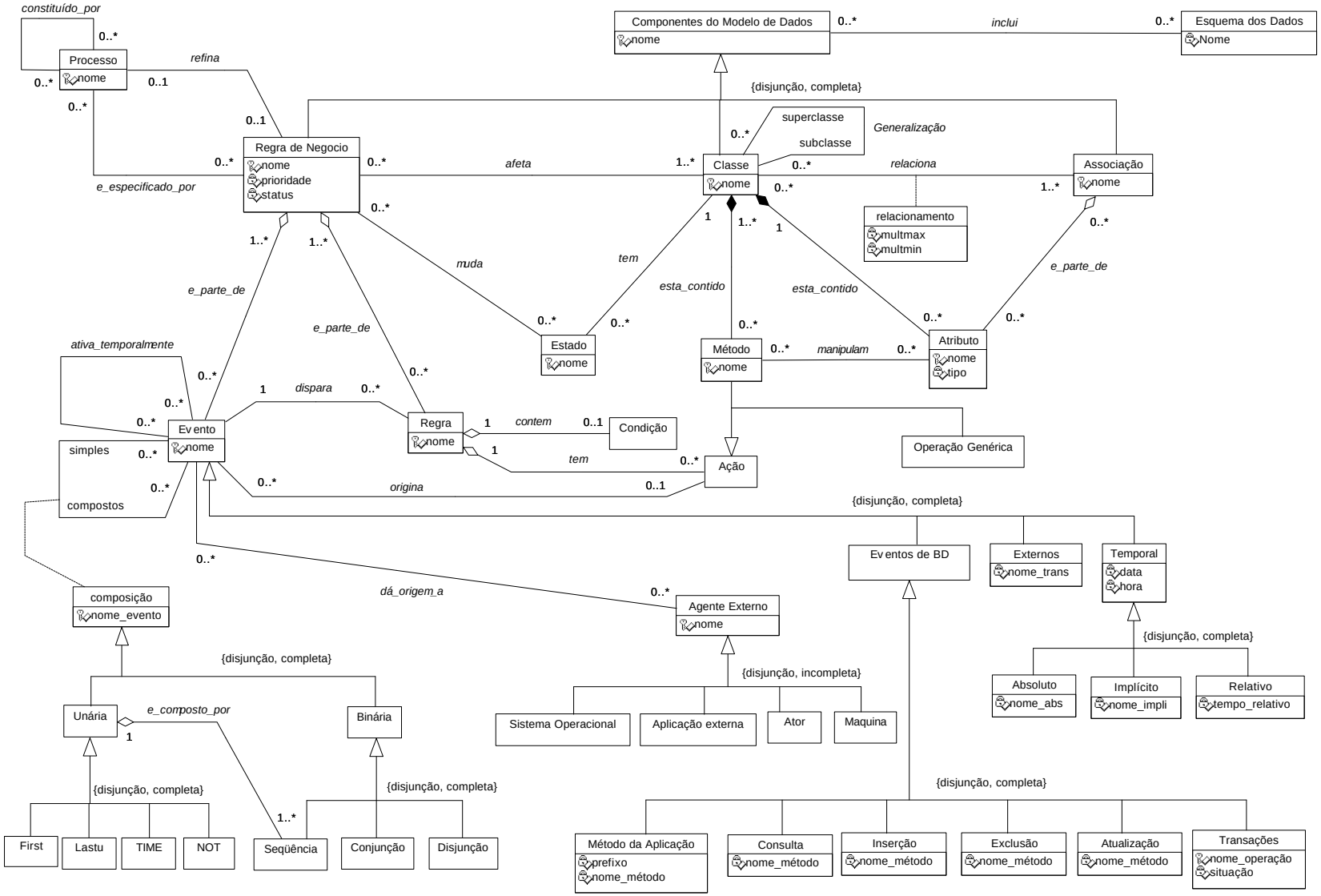

Figura 4: Metamodelo para regras de negócio

A MetaClasse *Evento Externo* é definida por usuários ou por programas de aplicação onde todas as atualizações devem ser visíveis ao mundo externo. Espera-se que tais eventos sejam descobertos fora do sistema. O evento externo possui o atributo *nome_trans* que é assinalado no sistema junto aos parâmetros que os acompanham.

A MetaClasse *Evento Temporal* é especificada através de um ponto único na linha do tempo. O evento temporal possui dois atributos próprios *data* e *hora* cujas características particulares da ocorrência de classe são: ano, mês, dia, hora, minuto e segundo. Para um melhor entendimento dos eventos temporais, estes se dividem em eventos temporais absolutos, implícitos e relativos.

Os *Eventos Compostos* são construídos a partir de construtores de eventos. O conjunto de construtores de eventos forma a álgebra do evento. Um evento composto é uma expressão regular construída a partir de eventos simples utilizando construtores de eventos unários e binários.

Os construtores binários de eventos são a conjunção (E_1, E_2), seqüência (E_1; E_2), e disjunção ($E_1|E_2$). Os construtores unários de eventos estão compostos sobre a base de um intervalo especifico I, mesmo que esse seja um seqüência de eventos E. Estes construtores são: first(F), last(L), TIME, NOT.

4.1.1.2 Regra

Esta classe é parte agregada da MetaClasse "Regra de Negócio". Por sua vez, é também um agregado de duas MetaClasses: "Condição" e "Ação", conforme descritas a seguir:

- Condição

  Toda regra pode ou não conter uma MetaClasse "Condição". A condição neste caso especifica um predicado a ser avaliado quando a regra é disparada e antes que a ação seja executada. Uma regra sem condição significa que a ação correspondente é executada incondicionalmente toda vez que o evento associado ocorre. A vantagem de se utilizar uma consulta para testar uma condição, ao invés de uma expressão booleana , é que o resultado da consulta pode ser utilizado posteriormente.

  O Metamodelo adiciona um operador Booleano para um fácil suporte da entrada de regras num projeto. Especificamente, adicionou-se uma estrutura "*IF-THEN-ELSE*" (Se-Então-Senão) para apoiar as restrições condicionais (descritas acima).

- Ação

  Toda regra pode ter ou não uma MetaClasse "Ação". A ação de uma regra é executada quando a regra é disparada e sua condição é verdadeira. Uma ação corresponde a uma seqüência de operações que podem ser codificadas por um método. Ações podem ser chamadas de procedimentos externos, bem como uma seqüência de operações de definição (funções) e manipulação sobre objetos. Está associada, ainda, a dois tipos de MetaClasses: "Evento" e "Ação". Conforme descritas a seguir:

  1. Cada ação pode *originar* mais de um evento.
  2. Ações estão *contidas* em uma regra

  A MetaClasse "Ação" possui duas subclasses:

  - Operação Genérica
    Corresponde a um comportamento resultante de um procedimento algorítmico. Também entende-se como algo invocado por um objeto (instância). As operações genéricas também são chamadas de funções ou procedimentos.
  - Método
    Uma ação poder ser também um método (componente da MetaClasse Classe, descrito a seguir).

4.1.2 Classe

A MetaClasse Classe é composta pelas MetaClasses “Método” e “Atributo”. Além disso se auto-relaciona com ela mesma e relaciona-se, ainda, com outras três MetaClasses: “Regra de negócio”, “Associação” e “Estado”.

Uma classe representa uma imagem para um grupo arbitrário de instâncias relacionadas, mas não necessariamente idênticas, segundo a descrição:

1. Uma Classe pode ser *afetada* por um conjunto finito de regras de negócio.
2. Uma classe *possui* um conjunto finito de estados.
3. As classes num diagrama de classes estão *relacionadas* através de associações.
4. Uma Classe pode especializar-se em uma ou mais subclasses.

Seus componentes agregados são descritos a seguir:

4.1.2.1 Atributo

Atributo é a menor unidade de informação que qualifica a propriedade de uma classe. A MetaClasse Atributo é parte, também, da MetaClasse Associação. Possui dois atributos: nome e tipo, que identificam o atributo e especificam o seu domínio.

4.1.2.2 Método

Um método corresponde a um corpo de procedimento e à implementação de uma operação, sendo identificado por um nome. A MetaClasse Método é responsavel pelo envio de mensagens aos objetos (instâncias) que *manipulam* os atributos da classe, semelhantemente à chamada de procedimentos em linguagens de programação convencionais.

4.1.3 Associação

A MetaClasse Associação, identificada por um nome representa um relacionamento que define uma conexão semântica entre pares de objetos. Sua semântica é expressa por:

1. Uma associação *relaciona* classes não correlatas.
2. A MetaAssociação “relacionamento” representa uma MetaClasse Associativa que *possui* atributos próprios que são: multiplicidade máxima (multmax) e multiplicidade mínima (multmin), onde multiplicidade representa a cardinalidade de uma associação.

Uma associação é constituída de zero ou mais atributos.

4.2 Esquema de dados

Num modelo de dados, é importante fazer a distinção entre a descrição do banco de dados e o próprio banco de dados. A descrição é conhecida como MetaClasse “Esquema de Dados” (ou metadados). Por sua natureza, ele inclui construções estruturadas orientadas ao armazenamento, tais como as definições dos métodos físicos de acesso. Além disso, é relacionada com a MetaClasse “Componentes do Modelo de Dados”, isto é, os esquemas de dados *incluem* um conjunto finito de componentes de modelos de dados.

4.3 Processos

A MetaClasse “Processo” é um transformador de informação que reside dentro dos limites do sistema a ser modelado. É um elemento primordial das organizações e das abordagens organizacionais como na reengenharia dos processos de negócio. Assim, para construir um modelo adequado do mundo real, eles devem ser enfatizados na modelagem conceitual. No Metamodelo existem três MetaAssociações que relacionam as MetaClasses “Processos” e as “Regras de Negócio” (uma MetaClasse componente de modelo de dados, descrito a seguir):

1. Cada processo *é especificado por* um conjunto finito de regras de negócio.

2. Um processo pode *refinar* uma regra de negócio.
3. Um processo está *constituído* por outros processos.

4.4 Estado

Um objeto é a instância de uma classe, que possui uma identidade embutida, independente de sua classe ou estado. Baseado nessa definição, todo objeto tem uma MetaClasse Estado (valor de suas variáveis de instância), assim como um estado representa uma condição ou a situação daquele objeto. Este satisfaz a alguma condição, executa alguma atividade em resposta a um evento, ou espera pela ocorrência de algum evento.

O atributo nome identifica o estado daquele objeto, e deve ser instanciado com um verbo no gerúndio ou particípio passado, para melhor expressar a sua ação.

4.5 Agente Externo

Essa MetaClasse abrange as seguintes MetaClasses: "Sistema Operacional", "Aplicação Externa", "Ator", "Máquina", etc., que podem dar origem a um conjunto finito de eventos. Um agente exterior como:

4.5.1 Sistema Operacional

Essa MetaClasse dá origem a eventos através de recursos de hardware, gerenciamento das interfaces entre a estação e os recursos externos, compartilhamento de recursos em rede, comunicação com outros sistemas operacionais.

4.5.2 Aplicação Externa

Essa MetaClasse dá origem a eventos através de programas de aplicação em componentes de software externos, desenvolvidos para o atendimento de uma necessidade específica de pessoas ou organizações. Pode ser também um produto padronizado (adquirido pronto para o uso).

4.5.3 Ator

Essa MetaClasse dá origem a eventos através de estereótipos predefinidos de tipo do usuário, denotando um agente fora do sistema que interage em diferentes casos de uso.

4.5.4 Máquina

Essa MetaClasse dá origem a eventos através de equipamentos e alarmes analógicos.

4.6 Modelo $(ER)^2$ Estendido para SGBDRO

O modelo $(ER)^2$ estendido (Entidade-Relacionamento-Evento-Regra com recursos de Orientação a Objetos) é uma extensão ao modelo $(ER)^2$ descrito na seção 3, que além de todas as características próprias do modelo, permite especificar:

- o comportamento ativo dos objetos na forma de evento e regra,
- o comportamento interno de um objeto através do conceito de estado,
- a execução das ações quando um método ou operação genérica é ativado, e
- a execução de um conjunto de ações em um objeto, e como este afeta outros objetos com os quais está relacionado.

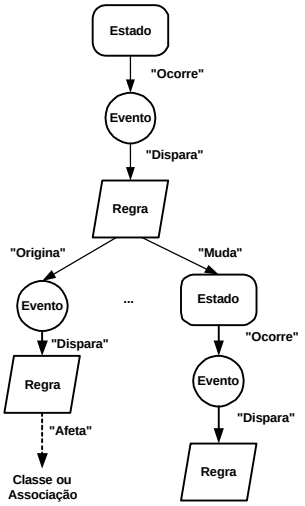

Figura 5: Representação diagramática de um comportamento ativo no $(ER)^2$ estendido.

O modelo $(ER)^2$ estendido (Figura 5) incorpora ao diagrama de classes e objetos a especificação de regras de negócio como objetos de primeira classe, tratando assim estados, eventos e regras como objetos do sistema.

As conexões que, num diagrama de transição de estado, ligam um estado a outro, como no diagrama da Figura 5, representam as diversas ligações entre os componentes estado, evento e regra. As conexões intuitivas são descritas a seguir:

- "Ocorre" expressa a ocorrência dos eventos/regras no próprio estado;
- "Dispara" indica que a ocorrência de um evento pode disparar um número finito de regras e também dar origem a outros eventos, como no caso de eventos temporais;
- "Origina" significa que a execução de uma regra pode originar diversos eventos;
- "Muda" expressa que a execução de uma regra pode causar a mudança de estado;
- "Afeta" liga a ação de uma regra a objetos de dados (classe, associação), e são rotuladas num diagrama $(ER)^2$ estendido como um tipo de método ou operação genérica que causa o evento (inserção, alteração, exclusão ou consulta).

Para entender esse comportamento como redes de Petri de alto nível, a condição ou situação do objeto representa o estado, lugares representam os eventos e transições representam o componente de ação da regra (Figura 6).

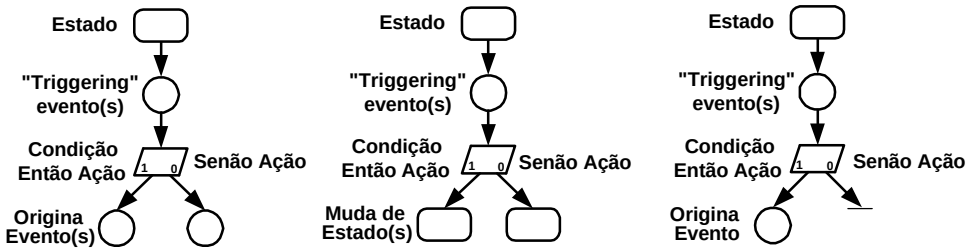

Figura 6: Comportamento ativo visto como rede de Petri de alto nível.

5. Conclusões

Neste artigo, constata-se a tendência de incorporar características do paradigma de orientação a objetos nos sistemas de informação baseados em bancos de dados, através dos SGBDs relacionais-objeto. Embora não implementem todas as características da orientação a objetos, eles acrescentam uma camada de abstração de dados sobre o modelo relacional, permitindo assim a manipulação de estruturas de dados mais complexas.

Por outro lado, esta nova tecnologia possibilita, também, a criação de ambientes de desenvolvimento baseados em regras de negócio, que interagem com os objetos do minimundo através da utilização de novos construtores para a modelagem de estados, eventos (com suas diferentes semânticas) e regras, que englobam os seus componentes intrínsecos de condição e ação. Esses construtores podem, então, ser mapeados para os comandos SQL que implementam essas novas características.

Para uma utilização eficaz de tais ambientes de desenvolvimento e da tecnologia gerada, existe a necessidade de técnicas de modelagem e ferramentas que facilitem a construção, alteração, consulta e depuração de esquemas de bancos de dados ativos. O modelo $(ER)^2$ estendido, conforme definido no capítulo 4, resolve semanticamente e graficamente esse problema.

Com o objetivo de comprovar a funcionalidade dessa abordagem, foi implementado um protótipo de ferramenta de projeto de regras de negócio. Pretende-se dar continuidade a esse trabalho, adicionando à ferramenta a capacidade de realizar integração de regras de negócio em esquemas provenientes da engenharia reversa de diferentes sistemas, tornando o ambiente adequado para todo o ciclo da reengenharia de sistemas de informação.

REFERÊNCIAS

[1] Amorim, M. J.; Ambiente de Desenvolvimento de Aplicações de Banco de Dados Ativos, Tese de Mestrado, IME, Rio de Janeiro, Dezembro 1998.

[2] Appleton, D.S.; Business Reengineering with Business Rules, in: V. Grover, W.J. Kettinger (Eds.), Business Process Change – Reengineering Concepts, Methods and Tecnologies, Londres, Harrisburg 1995.

[3] Bell, J., Brooks, D., Goldbloom, E., Sarro, R., Wood, J.; Re-Engineering Case Study – Analysis of Business Rules and Recomendations for Treatment of Rules in a Relational Database Environment, Bellevue Golden: US West Information Technologies Group 1990.

[4] Booch, G.; Rumbaugh, J.; Jacobson, I,; "The Unified Modeling Language for Object-Oriented Development", Documentation Set Version, janeiro, 1997.

[5] Chakravarthy, S., Krishnaprasad, V., Anwar, E., Kim, S-K.; Composite Events for Active Databases: Semantic Context and detection; Proceeding of the 20th Conference on Very Large Databases, Los Alamitos: IEEE Computer Society Press, 1994.

[6] Costa, A.; Engenharia Reversa de Regras Ativas em Banco de Dados Relacionais, Tese de Mestrado, IME, Rio de Janeiro, Dezembro, 1997.

[7] Fäberböck, H., Gutzwiller, T., Heym, M.; Ein Vergleich von Requeriments Engineering Methoden auf Metamodell-Basis, in: M. Timm(Ed.), Requirements Engineering, Springer, Berlin, 1991.

[8] Gatziu, S.; Events in an Active Objcet-Oriented Database System, Hamburg, Kovac, 1994.

[9] Herbst, H.; Business Rule-Oriented Conceptual Modeling, Physica-Verlag: A Springer-Verlag Company, Switzerland, Bern, November 1996.

[10] Herbst, H., Knolmayer, G.; Ansätze zur Klassifikation von Geschäftsregeln, in: Wirtschaftsinformatik, 1995.

[11] Hsu, M., Cheatham, T.E.Jr.; Rule Execution in CPLEX: A Persistent Objectbase, in: K.R. Dittrich (Ed.), Advances in Object-Oriented Database Systems, Berlin, Springer, 1988.

[12] Loucopoulos, P., Conceptual Modeling, in: P. Loucopoulos (Ed.), Conceptual Modeling, Databases, and CASE, New York et. al.: John Wiley & Sons 1992.

[13] MOF, Meta Object Facility; Object Management Group; Specification: Joint Revised Submission Cooperative Research Centre for Distributed System Technology (DSTC); IBM Corporation, International Computers Limited, Object Inc, Oracle Corporation, System Software Associates, Unisys Corporation; http://www.omg.org/techprocess/meetings/schedule/techtab.html#mof, OMG document, september, 1997.

[14] Tanaka, A.; On Conceptual Design of Active Database, Ph Thesis; Georgia Institute of Tecnology, 1992.

[15] Widom, J., Ceri, S.; Active Database Systems Triggers and Rules for Advanced Database processing: Morgan faufmann Publishers, Inc., San Francisco California, 1996.