UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

EXECUÇÃO DE SERVIÇOS BASEADA EM REGRAS DE NEGÓCIO

Aqueo Kamada

Orientador: Manuel de Jesus Mendes

Tese de Doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Doutor em Engenharia Elétrica. Área de concentração: Engenharia de Computação.

Campinas, SP

2006

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE -
UNICAMP

Kamada, Aqueo
K127e Execução de serviços baseada em regras de negócio / Aqueo Kamada. --Campinas, SP: [s.n.], 2006.

Orientador: Manuel de Jesus Mendes
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Serviços na Web. 2. Engenharia de software. 3. Ontologia. 4. Software – Desenvolvimento. 5. Middleware. I. Mendes, Manuel de Jesus. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Service execution based on business rules
Palavras-chave em Inglês: Business rule, Business rule engine, SBVR metamodel, SOA, Process composition, Ontology
Área de concentração: Engenharia de Computação
Titulação: Doutor em Engenharia Elétrica
Banca examinadora: Eleri Cardozo, Fabrício Alves Barbosa da Silva, Ivan Luiz Marques Ricarte e Juan Manuel Adan Coelho
Data da defesa: 12/12/2006
Programa de Pós-Graduação: Engenharia Elétrica

Aqueo Kamada

Execução de Serviços baseada em Regras de Negócio

Tese de Doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Doutor em Engenharia Elétrica. Área de concentração: Engenharia de Computação.
Aprovação em 12/12/2006

Banca Examinadora:
Prof. Dr. Manuel de Jesus Mendes – UNICAMP
Prof. Dr. Eleri Cardozo – UNICAMP
Prof. Dr. Fabrício Alves Barbosa da Silva – Unisantos
Prof. Dr. Ivan Luiz Marques Ricarte – UNICAMP
Prof. Dr. Juan Manuel Adan Coello – PUC-Campinas

Campinas, SP
2006

Resumo

No atual mundo globalizado, com interações fortemente baseadas na Web, as relações entre pessoas, empresas e órgãos de governo estão sujeitas a mudanças cada vez mais rápidas. Esse cenário torna ainda mais crítico o já antigo problema de manutenção de sistemas, uma vez que as atualizações precisam ser realizadas para contemplar novos modelos de negócio, no contexto destas mudanças rápidas.

Esta situação justifica a necessidade de uma abordagem que consiga capturar as mudanças nos negócios e rapidamente implementá-las nos sistemas computacionais. Considera-se que existem porções da lógica de negócio que são bastante voláteis e suscetíveis às mudanças e outras porções que são bastante estáveis e muito pouco suscetíveis às mudanças. Neste contexto, propõe-se um modelo para desenvolvimento rápido e execução de serviços, baseado em regras de negócio. Neste modelo as porções voláteis são externalizadas como regras de negócio e as porções estáveis como serviços.

A prova de conceito desta abordagem traduz regras modeladas em linguagem natural para regras no modelo da máquina de regras instanciada, que invoca os serviços.

Palavras-chave: Regra de Negócio, Máquina de Regras, metamodelo SBVR, SOA, processo de composição, Ontologia.

Abstract

In the current globalized world, characterized by strong Web based interactions, relationships between people, companies and government organizations are subject to changes that occur faster and faster. This scenario makes the old problem of system maintenance even more critical, as the updating of these systems needs to contemplate new business models in the context of these fast changes.

This situation justifies the need for an approach that can capture business changes and quickly implement them into computational systems. It is considered that some business logic portions are quite volatile and susceptible to the changes and other portions are quite stable and less susceptible to the changes. In this context, a model for fast development and execution of services based on business rules is proposed. In this model the volatile portions are externalized as business rules and the stable portions as services.

The proof of concept of this approach translates rules which are modeled in a natural language to rules in the model of the instantiated rule engine that invokes the services.

Keywords: Business Rule, Business Rule Engine, SBVR metamodel, SOA, Composition Process, Ontology.

Agradecimentos

Agradeço ao meu orientador Professor Manuel de Jesus Mendes pelo constante estímulo, pelo exercício da paciência possível diante do meu estilo de personalidade. Trabalhar nesta tese sob sua orientação foi uma experiência muito rica e foi fundamental para que este trabalho chegasse ao ponto em que se encontra. Agradeço pelas ponderações nos momentos em que tentei "viajar" e por me incentivar quando eu teimava ou não percebia a necessidade de "voar alto".

Institucionalmente agradeço ao CenPRA – Centro de Pesquisas Renato Archer e em especial à DSSD - Divisão de Software para Sistemas Distribuídos pelo apoio à capacitação do seu corpo técnico. Agradeço também à Unicamp e ao projeto eGOIA pelo suporte financeiro à participação em congressos.

Agradeço ao Marcos Antonio Rodrigues, chefe da DSSD, pelo incentivo a este trabalho e apoio ao meu afastamento para finalizar esta tese. Agradeço à toda a equipe do CenPRA no projeto eGOIA. Direta ou indiretamente todos contribuíram com este trabalho, na medida em que muitas das idéias discutidas no projeto serviram como ponto de partida para a amplificação dos conceitos e modelos desenvolvidos ou foram adaptadas para o cenário usado na prova de conceito desta tese. Ao Walcir pelas dicas sobre ferramentas de Serviços Web e ao Rodrigo sobre o uso da máquina de regras Jess. À Adriana e ao Gonzaga, sempre amigos, pelas dicas e ajudas de todas as horas.

Pelo lado mais afetivo agradeço à minha querida esposa Fátima pelo amor e apoio sempre presente e incondicional e aos meus queridos filhos Marcela, Diogo, Frederico e Hugo pela alegria que têm me dado, mesmo estando, três deles, do outro lado do mundo. Eu gostaria de expressar minha gratidão à Elza e à Odete (queridas cunhadas) e à minha sobrinha Camila pela compreensão nos momentos em que estive ausente e mesmo assim deram apoio sempre presente e incondicional nesta minha jornada. Odete, obrigado também pelas incontáveis caronas! À minha mãe Shizuka e meu saudoso pai Kesao por tudo de bom que me deram. Aos meus irmãos Yutaka, Celso e Toshio e irmãs Satiko e Setsuko (Nem) que apoiaram para que eu desse os meus primeiros passos para a minha vida.

Minha sincera gratidão a todos eles!

À minha querida esposa
Fátima
e aos meus queridos filhos
Marcela, Diogo, Frederico e Hugo,
com carinho, ternura e amor.

Sumário

Lista de Figuras

Lista de Tabelas

Glossário

Termo ou Sigla	Descrição
AOP	*Aspect Oriented Programming*
Binding	Especifica um formato de mensagem e detalhes de protocolo para operações e dados de serviço num ponto de interação particular. Tais informações de ligação são necessárias numa especificação de serviço para estabelecer exatamente como os consumidores e os provedores do serviço conectam-se entre si usando os canais de serviço num ambiente distribuído e heterogêneo.
BMM	*Business Motivation Model* é uma especificação a ser adotada pela OMG para definir 'por que' uma organização deve ter suas regras de negócio para governar suas ações, ou seja, para determinar o que deve fazer com seus produtos e serviços, suas pessoas, locais, e tempo.
BPDM	*Business Process Definition Metamodel*
BPEL	*Business Process Execution Language*
BPMN	*Business Process Modeling Notation*
BRT	*Business Rules Team* – Consórcio, formado especificamente para responder ao BSBR RFP, que produziu o metamodelo SBVR.
BSBR RFP	*Business Semantic of Business Rules – Request for Proposal* – Solicitação de proposta da OMG para metamodelagem de regras de negócio. Freqüentemente abreviado como BSBR.
Canal de serviço	Representa uma conexão ou comunicação entre dois pontos de interação de serviço. Estabelece uma dependência entre os pontos de interação do consumidor e do provedor.
CIM	*Computation Independent Model* é a visão de um sistema do ponto de vista independente de computação e, como tal, mostra o modelo de negócio de um sistema do ponto de vista de um analista de negócio. Um CIM não mostra detalhes da estrutura de sistemas.
Componente de serviço (ou serviço Web)	Uma "unidade de trabalho" que realiza uma tarefa, no âmbito das relações entre processos computacionais. Esta "unidade de trabalho" é realizada por um "provedor de serviço" para satisfazer necessidades de um "consumidor de serviço", de acordo com um "contrato", que pode opcionalmente ser através de "intermediador de serviço". Note que um componente de serviço embora também possa realizar um serviço para pessoas, de modo geral, apresenta um nível de granularidade menor do que o de um serviço.
CORBA	*Common Object Request Broker Architecture*
DAML	*DARPA Agent Markup Language* é um programa que oficialmente

	começou em agosto de 2000 com a meta de desenvolver uma linguagem e ferramentas para facilitar a conceituação da Web Semântica.
DAML+OIL	DAML+OIL é uma linguagem de marcação semântica para recursos na Web, baseada nos padrões RDF e RDF Schema, para estender estas linguagens com primitivas semanticamente mais ricas.
EDOC	*Enterprise Distributed Object Computing*
Encadeamento progressivo	Uma classe de algoritmos para executar regras que usa encadeamento progressivo (*Forward Chaining*). Também, conhecido como encadeamento dirigido por dados, parte de fatos conhecidos e tenta deduzir novos fatos, até chegar à solução.
Encadeamento regressivo	Uma classe de algoritmos que usa encadeamento regressivo (*Backward Chaining*) para determinar quais regras se aplicam para testar uma hipótese. Encadeamento regressivo pode ser definido como: "Um algoritmo para provar uma meta pela quebra recursiva da meta em sub-metas e tentar prová-las até que os fatos sejam verdadeiros. Fato é uma meta sem sub-meta que é então considerado sempre verdadeiro."
IDE	Integrated Development Environment
JSR 94	Java Specification Request - JSR 94: Java™ Rule Engine API. Define uma API Java para máquina de regras
Lógica de Primeira Ordem	Nessa lógica o universo de discurso é composto por objetos que possuem certas propriedades e que relacionam-se entre si. É chamada de primeira ordem porque estes objetos representam as entidades de "primeira ordem" do universo de discurso e as sentenças expressam declarações sobre propriedades e relações entre objetos individualizados (especialização) ou entre os conjuntos de objetos de classes (generalização).
Lógica deôntica	É a disciplina que investiga os princípios da argumentação válida. Em particular, a lógica deôntica estuda a validade de argumentos nos quais frases regidas por expressões como *É obrigatório que...*, *É permitido que...* desempenham papel relevante. A lógica deôntica tem seu nome derivado da palavra grega *déon* (necessidade, o que é preciso). Essa lógica pode ser entendida como a lógica das normas, no sentido do que seja obrigatório ou permitido.
MDA	*Model Driven Architecture* - baseado em UML, MOF e CWM, propõe separar a arquitetura de uma aplicação de sua implementação, de modo que a arquitetura da aplicação não fique amarrada a tecnologias de implementação. MDA é um abordagem ou *framework* com foco no uso de modelos no desenvolvimento de software, diferente de OMA e CORBA, que são *frameworks* para implementar sistemas distribuídos.
Mecanismo ou Máquina de regras de	Máquina virtual que executa regras de negócio armazenadas na sua memória de trabalho. De modo geral, mecanismos de regras referem-se a qualquer software que executa regras e neste sentido as

negócio	chamadas máquinas de inferência são também mecanismos de regras. Estes mecanismos provêm um algoritmo ou um conjunto de algoritmos, como encadeamento progressivo ou regressivo para executar as regras.
MOF	*Meta Object Facility* é uma especificação onde modelos podem ser exportados de uma aplicação, importados em outra, transportados em uma rede, armazenados em repositórios e então recuperados em diferentes formatos (incluindo XMI), transformados, e usados para geração de código executável.
OCL	*Object Constraint Language*
ODM	*Ontology Definition Metamodel*
OMG	*Object Management Group* - OMG é um consórcio de padronização em TI, mais conhecido por seus esforços em padrões, como UML e CORBA.
OWL	*Web Ontology Language*
PIM	*Platform Independent Model* é uma visão de um sistema do ponto de vista independente da plataforma. Um PIM especifica um grau de independência de plataforma de modo que possa ser aplicado em diferentes plataformas.
Ponto de Interação	É um ponto de interação ou porta através da qual os consumidores de um serviço conectam-se com os provedores nos canais de serviço (service channel). Pontos de interação tratam das dependências entre consumidores e provedores de serviço minimizando o acoplamento entre eles em relação aos recursos necessários numa interação em particular e separando-as de outras interações.
Proxy	Programa cliente que representa o serviço remoto.
PRR	*Production Rule Representation*
PSM	*Platform Specific Model* é uma visão de um sistema do ponto de vista de uma plataforma específica. Um PSM combina as especificações no PIM com os detalhes que especificam como aquele sistema usa um tipo particular de plataforma.
RAD	*Rapid Application Development*
RDF	*Resource Description Framework* é uma linguagem de propósito geral para representar informação na Web. Define um modelo simples para descrever relações entre recursos em termos de nomes e valores de propriedades. Baseia-se em triplas (Sujeito, Verbo e Objeto). Uma declaração é uma tripla RDF instanciada.
RDFS	RDF Schema descreve como usar RDF para descrever vocabulários RDF. Define também um vocabulário básico para este propósito e provê um mecanismo para descrever propriedades e as relações entre elas e outros recursos. RDF Schema define classes e propriedades que podem ser usadas para descrever outras classes e propriedades.
Regra de produção	É uma declaração independente da lógica de programação na forma "SE <condição> ENTÃO <ação>" que é executável por um mecanismo de regras.

RETE	É um algoritmo de execução de regras muito rápido e, por isso, um dos mais usados, que cria uma rede e ao invés de buscar as semelhanças dos fatos em todas as regras em todos os ciclos de reconhecimento das ações, procura buscar somente por mudanças nos fatos semelhantes em todos os ciclos.
SBVR	*Semantics of Business Vocabulary and Business Rules*.
Serviço	Uma unidade de trabalho que produz um valor de fácil assimilação para as pessoas no âmbito dos negócios e das relações entre pessoas, empresas e governos.
SOA	*Service Oriented Architecture*
SW	Serviço Web
SWRL	*Semantic Web Rule Language*
UDDI	*Universal Description, Discovery, and Integration*
UML	*Unified Modeling Language*
UML OCL & AS	*UML - Object Constraint Language & Action Semantics*
URI	*Universal Resource Identifier*
URL	*Universal Resource Locator*
Vocabulário de negócio	Conjunto estruturado de termos e símbolos juntamente com seus significados e relações entre eles, para uso de uma comunidade de negócio.
W3C	*World Wide Web Consortium*
WS-BPEL	*Web Services Business Process Execution Language*
WSDL	*Web Services Description Language*
XMI	*XML Metadata Interchange*
XML	*eXtensible Markup Language*
XML Schema	*eXtensible Markup Language Schema*

Trabalhos Publicados Pelo Autor

1. A. Kamada e M. J. Mendes. “A Business Rule Engine Applied to eGovernment Services Integration”. *5th IFIP Conference e-Commerce, e-Business, and e-Government (I3E´2005),* Poznan - Polônia - Outubro/2005. Disponível também no livro “Challenges of Expanding Internet: E-Commerce, E-Business, and Egovernment”. Springer Verlag New York Inc, 2005, ISBN: 0387287531, p. 157-171.
2. A. Kamada e M. J. Mendes. “Rule based flexible eGovernment applications”. *Proceedings of the International Conference on e-Government (ICEG 2005),* Ottawa - Canadá - Outubro/2005.
3. N. Tizzo, A. Kamada, A. M. C. M. Figueiredo, M. J. MENDES, J. G. Souza Jr, M. A. Rodrigues, L. Damasceno, J. R. Borelli. “Service Composition Applied to e-Government”. *IFIP WCC 2004 - 4th IFIP Conference e-Commerce, e-Business, and e-Government (I3E´2004),* Toulouse - França - Agosto/2004. Disponível também no livro “BUILDING THE E-SERVICE SOCIETY E-Commerce, E-Business, and Egovernment”. Kluwer Academic Publishers, Agosto/2004, ISBN: 1402081545, p. 307-326.
4. A. M. C. M. Figueiredo, A. Kamada, M.J. Mendes, M. A. Rodrigues, L. Damasceno. “Metadata Repository Support for Legacy Knowledge Discovery in Public Administrations”. *KMGov2004 - 5th Working Conference on Knowledge Management in electronic Government*, Krems - Austria - Maio/2004 Disponível também no livro “Knowledge Management in Electronic Government”, Lecture Notes in Computer Science Springer 2004, ISBN 3-540-22002-X, v. 3035, p. 157-165.
5. A. M. C. M. Figueiredo, A. Kamada, M. J. Mendes, M. A. Rodrigues, J. G. Souza Jr., L. Damasceno, J. R. Borelli, N. Tizzo. “Applying MDA Concepts in an e-Government Platform”. *Proceedings of the 3rd International Conference on Electronic Government - EGOV '04*, Zaragoza, Espanha, 2004. p. 260-266.

6. J. L. Cardoso Jr., A. Kamada, A. M. C. M. Figueiredo, M. J. Mendes,; P.Hoepner, ; R. Monte,; S. Bolliger. "Implementing Electronic Government: The eGOIA Project". *EU-LAT Workshop on e-Government and e-Democracy: Progress and Challenges,* Santiago - Chile - Maio/2004.
7. A. M. C. M. Figueiredo, A. Kamada, M.J. Mendes, M. A. Rodrigues, L. Damasceno. "Using metamodels to promote Data Integration in e-Government Application Scenario". *3rd IFIP I3E2003 Conference on eCommerce, eBusiness and eGovernement* - Guarujá - São Paulo - Brazil - Setembro/2003. Disponível também no livro "Digital Communities in a Networked Society - e-Commerce, e-Business and e-Government ", Kluwer Academic Publishers, 2004, ISBN: 1402077955, v. 139, p. 293-303.
8. M. J. Mendes, A. Kamada, A. M. C. M. Figueiredo, M. A. Rodrigues, L. Damasceno, J. R. Borelli, N. Tizzo, C.R. Muniz. "Federated Web Services". *Workshop - Sistemas Distribuídos*, Copiapo – Chile. Novembro/2002.
9. W. Meng, A. Kamada, Y-H. Chang "Transformation of Relational Schema to Object-Oriented Schema". *Proc. of the Nineteenth Annual International Computer Software and Applications Conference (COMPSAC'95)*, pp.356-361, Dallas, TX, E.U.A, Agosto/1995.
10. A. Kamada. "ADABAG Baguete Application Development Environment". *16th Annual Computer Science Conference of Federation of North Texas Area Universities*. Dallas, TX, EUA, 1992.

* Desta lista de trabalhos publicados os de 1 a 8 estão relacionados com esta tese.

Capítulo 1

Introdução

Neste capítulo apresenta-se uma visão geral das tecnologias de regras de negócio, na implementação e manutenção de sistemas computacionais, inseridas no contexto mundial dos negócios via Web. As motivações, o objetivo e as contribuições deste trabalho são brevemente discutidas à luz das deficiências, fraquezas e complexidades associadas às tecnologias envolvidas na manutenção de sistemas.

1.1 Visão Geral sobre Regras de Negócio

O comércio eletrônico cresceu rapidamente nos últimos anos e trouxe consigo muitas mudanças no comportamento dos negócios, mercados e consumidores. Os agentes que operam esse rápido movimento para uma economia e sociedade de comércio eletrônico são empresas comerciais já estabelecidas ou novas, que surgem aproveitando oportunidades antes inexistentes. Cada vez mais sistemas B2B (*Business to Business*) e B2C (*Business to Customer*) são estabelecidos para facilitar os negócios e transações *on-line*. Na esteira desse crescimento, surgiram variantes desse modelo de negócio, conduzidas por novos agentes, organizações públicas e privadas, onde são estabelecidos sistemas, tais como os categorizados como G2G (*Government to Government*), G2C (*Government to Citizen*) e B2G (*Business to Government*).

Em geral, um negócio é suportado por um sistema computacional que é usado para ajudar a organizar os dados e a administrar as atividades cotidianas de um negócio. Conseqüentemente, um sistema computacional deve refletir o conhecimento da área de negócio na qual o sistema é usado. As estratégias, as políticas e as relações entre as organizações consolidam esse conhecimento e estão definidas, via de regra, em documentos, tais como procedimentos, contratos, regulamentações e leis. A partir destes

documentos, que podem ser internos ou externos às organizações, originam-se as regras que definem o comportamento dos negócios destas organizações **[1]**. Em outras palavras, tais regras, conhecidas como regras de negócio, que contêm informações importantes para a realização do negócio, confundem-se com a própria essência do negócio e definem como uma organização deve se comportar diante das situações rotineiras no contexto em que atua.

Em um sistema de negócio tradicional, os conhecimentos sobre o comportamento do mesmo estão embutidos em códigos computacionais, SGBDs (Sistemas Gerenciadores de Banco de dados), gatilhos (*trigers*) e procedimentos armazenados (*stored procedures*) **[2]**. Estas diferentes abordagens inevitavelmente misturam o processamento do conhecimento do negócio com outras funções de programação.

No cenário atual altamente globalizado, com as relações entre as pessoas e as organizações baseadas em aplicações computacionais via Web, a necessidade de mudanças nestas aplicações acontece em períodos cada vez mais curtos. Novas tecnologias, novas aplicações e novas necessidades alimentam a demanda por mudanças e por soluções que integrem rapidamente as aplicações existentes com as novas que vão surgindo a todo instante [3]. A necessidade de permitir variações em processos repetitivos automatizados, garantindo aderência a regulamentações complexas e voláteis aumenta a demanda por infra-estruturas baseadas em múltiplas tecnologias e em constante evolução para suportar o desenvolvimento rápido de novos sistemas computacionais.

Pessoas, governos e empresas, que esperam e pressionam por implementações quase instantâneas, estimulam esta demanda por mudanças. Esta alta demanda é conseqüência das mudanças que ocorrem nas regras de negócio.

1.2 Objetivo e Motivação

Este trabalho parte da premissa de que é necessário prover uma maneira de contemplar rapidamente nos sistemas computacionais as mudanças que ocorrem no dia-a-dia nos negócios das organizações. Assim, o objetivo é a proposição de um modelo flexível para desenvolvimento de regras de negócio e execução de serviços (no sentido lato), baseado em

regras de negócio, que consiga refletir rapidamente as mudanças nos sistemas computacionais das empresas e dos governos.

A motivação para a pesquisa deste trabalho de tese originou-se a partir da percepção das dificuldades ou inflexibilidade na manutenção de sistemas computacionais para refletir rapidamente as mudanças nas organizações. As razões identificadas para estas dificuldades estão listadas a seguir:

- Na grande maioria dos sistemas computacionais as regras de negócio estão espalhadas na documentação e no código executável e isto torna a manutenção demorada e cara.
- A ortogonalidade das perspectivas dos mecanismos de composição de processos e dos mecanismos de regras de negócio dificultam a integração dos mesmos no que se refere à colaboração para tratar da lógica do negócio.
- Os mecanismos de regras não conseguem transformar regras, descritas em linguagens amigáveis para analistas de negócio, em código computacional sem a ajuda de engenheiros de software.
- A falta de padronização nas linguagens de definição de regras, que são proprietárias na sua grande maioria, dificulta o intercâmbio de regras entre diferentes mecanismos de regras..

A seguir alguns detalhes para estas razões. A necessidade de mudanças nas aplicações computacionais ocorre devido às mudanças nas condições do mercado, às freqüentes associações e fusões entre organizações e às alterações contratuais e legais **[4]**. A necessidade de, concomitantemente, seguir regulamentações complexas e manter o impulso e a agilidade de um negócio desafia as organizações a buscarem melhores soluções para automatizar processos repetitivos e ao mesmo tempo permitir variações **[5]**. Assim, surge a demanda por infra-estruturas baseadas em tecnologias complexas para apoiar o desenvolvimento de uma grande gama de novos processos de negócio.

Considerando-se o estágio em que se encontra a evolução das infra-estruturas e tecnologias de desenvolvimento de sistemas, a tendência predominante é que, em muitos casos, estes novos processos proverão uma fachada integrada, orientada a serviços, para aplicações e dados legados. Em outros casos, serão desenvolvidas novas aplicações de

negócio também orientadas a serviços, para suportar funções criadas a partir da re-engenharia de sistemas. Em quaisquer destes casos, muito provavelmente os novos processos estarão baseados em Serviços Web **[6]**.

Nesse contexto, empresas e governos precisam incorporar novas tecnologias para implementar soluções que integrem processos e serviços já existentes com novos processos para gerar outros serviços, contemplando as novas necessidades. Desta forma, a cada instante, as regras de negócio, os processos e os serviços precisam ser revistos e realinhados para que a organização possa atuar de acordo com as mudanças que ocorrem no cenário em que se insere. Para a maioria das organizações a capacidade de adaptação a novas realidades de maneira rápida e eficiente é um desafio crucial para manter a sua competitividade ou a sua sobrevivência no mercado.

Nos últimos anos as pesquisas e as tecnologias relacionadas com regras de negócio ganharam um novo impulso devido ao seu potencial para capturar e facilitar a implementação das mudanças que ocorrem no dia-a-dia das organizações. Vários mecanismos de regras de negócio [7, 8, 9], os chamados *business rule engines*, bem estabelecidos, já demonstraram a efetividade de seu uso em alguns setores do mercado, notadamente nos setores de finanças e seguros. Os mecanismos de regras garantem flexibilidade ao possibilitar que potencialmente todos os pontos de decisão de fluxos de processos possam ser editados e armazenados em um repositório de regras gerenciado. Tradicionalmente, estes pontos de decisão são codificados na aplicação, mas com um mecanismo de regras, os pontos de decisão podem ser escritos na forma de regras de negócio. Assim, espera-se que as aplicações possam acessar os mecanismos de regras, e as próprias regras possam ser atualizadas rapidamente pelos analistas de negócio. Isto vai de encontro ao que sugeriu Barnes [10]: “a maioria das organizações que adquire mecanismos de regras quer que seus gerentes de negócio possam criar e editar suas próprias regras e, portanto, é importante que um mecanismo de regras forneça uma interface amigável para o pessoal não técnico.”. Os problemas mais comuns em todos os mecanismos de regras referem-se à dificuldade na formalização de regras em linguagem natural com respectivo suporte de execução e à dificuldade na combinação das mesmas com outros processos do negócio.

Por outro lado, as pesquisas e as tecnologias relacionadas com a arquitetura de sistemas baseada em serviços (SOA - Service Oriented Architecture) [11, 12] também ganharam forte impulso com a popularização dos Serviços Web [13] pelo seu potencial na integração e interoperação de serviços distribuídos, independentemente de localização, de plataforma de desenvolvimento, de ambiente de execução e de linguagem de programação [14]. Também nesta área os mecanismos de regras podem ter um papel importante [15]. As aplicações legadas ensejam fortes razões para que as organizações recorram aos mecanismos de regras e à tecnologia SOA. Quando as organizações têm muitas regras embutidas em aplicações legadas, migrá-las para um mecanismo de regras e manter os processos estáveis como componentes de serviço permite que os usuários façam mudanças sem terem de reescrever o código constantemente [3]. Aqui também, os problemas mais comuns referem-se à dificuldade na combinação das regras de negócio com os componentes de serviço encapsulando processos do negócio.

Recentemente as iniciativas de inclusão de mecanismos de regras de negócio no contexto das máquinas de composição de serviços tal como BPEL, têm contribuído para facilitar o processo de composição de serviços, embora falte a elas a habilidade para tratar adequadamente as regras de negócio de maneira colaborativa com a execução de serviços e no que se refere ao uso de regras de negócio tratadas por diferentes mecanismos de regras [16, 17]. Os mecanismos de regras de negócio estão despertando o interesse da comunidade de gerenciamento de processos de negócio (BPM - Business Process Management), conduzindo a parcerias entre fornecedores de BPM e fornecedores de mecanismos de regras de negócio [18].

1.3 Contribuições deste Trabalho

Em aplicações empresariais e de governo existem porções da lógica de negócio que são dinâmicas e sensíveis às mudanças do mercado e outras porções que são bastante estáveis. Porém, a forma monolítica como a grande maioria das aplicações foi desenvolvida e implementada com as porções dinâmicas embutidas no código computacional dificulta

imensamente a implementação de mudanças. Neste cenário, as tecnologias de regras de negócio voltaram a ganhar importância devido ao seu potencial de flexibilizar as mudanças.

Estas porções, dinâmicas e estáveis, precisam ser adequadamente estruturadas e mantidas separadas de modo que possam ser atualizadas de maneira independente. Depois estas porções dinâmicas e estáveis precisam ser integradas para concretizar novos serviços no contexto das mudanças rápidas que o mercado exige.

Para responder rapidamente a estas mudanças exigidas pelo mercado, este trabalho faz as seguintes contribuições, ao propor um modelo de desenvolvimento e execução de serviços composto de duas plataformas e de um conjunto de repositórios:

- Uma plataforma de desenvolvimento composta por um conjunto de ferramentas para suportar o desenvolvimento formal de regras na terminologia de pessoas de negócio.
- Uma plataforma de execução orientada a serviços composta por um preparador de execução e um executor de serviços baseado em mecanismos de regras.
- Os repositórios de vocabulários, regras e ontologias garantem o suporte necessário em termos de semântica, persistência e integridade para os artefatos manipulados, uma vez que podem ser usados concorrentemente pelas duas plataformas propostas.

O ambiente de desenvolvimento de regras de negócio proposto faz contribuições inéditas em relação a outras iniciativas semelhantes em regras de negócio. O ambiente dá suporte aos analistas de negócio na definição de regras, usando uma linguagem bastante familiar a eles, ou seja, usando os elementos e artefatos com os quais eles realizam seus negócios. A formalização de regras de negócio é baseada nas propostas do emergente metamodelo SBVR (Semantics of Business Vocabulary and Business Rules) [19].

A plataforma de execução orientada a serviços também apresenta características inéditas em relação a outras iniciativas semelhantes. As principais características que distinguem o módulo preparador de execução são que ele faz uso de regras de negócio para orientar a descoberta ou necessidade de realização de serviços e antecipa a disponibilidade de componentes de serviço valendo-se da estrutura ontológica do serviço que será executado. As principais características que distinguem o módulo executor de serviços são que ele (i) traduz em tempo de execução as regras de negócio formalizadas em linguagem natural para regras computacionalmente executáveis; (ii) faz a geração de programas-

cliente para os componentes de serviço em tempo de execução; e (iii) instancia e controla a execução de um mecanismo de regras que, por sua vez, invoca os componentes de serviço.

Esta proposta de modelo aproveita uma parte básica que é comum a muitas das propostas de arquiteturas de execução de regras de negócio [20]. O que não está presente na maioria das arquiteturas é a idéia em que se fundamenta a execução orientada a serviços. A orientação a serviços refere-se ao fato de que a necessidade de realização de um serviço (no nível do negócio) é o evento que ativa a execução do ambiente de execução.

A partir das contribuições indicadas pode-se extrair os seguintes aspectos que se mapeiam diretamente nas razões que motivam o trabalho e que ajudam a flexibilizar a manutenção de sistemas computacionais:

- O ambiente de desenvolvimento de regras contempla um método de externalização de regras de negócio da lógica das aplicações para facilitar a manutenção de sistemas computacionais.
- Na medida em que um processo de composição (no âmbito de mecanismos de composição, tais como o BPEL) é também um serviço então as vantagens dos mecanismos de regras de negócio podem ser aproveitadas tanto no ambiente de desenvolvimento no instante de ligação das regras com os serviços, quanto no ambiente de execução. Estas vantagens estão associadas à externalização das regras combinadas com as vantagens dos mecanismos de composição de serviços, no que se refere à provisão de alta disponibilidade e distributividade dos serviços.
- O Executor de serviços incorpora algoritmos de transformação automática de regras de negócio formais, descritas na terminologia das pessoas de negócio, para regras executáveis.
- Todos os repositórios incorporam modelos de representação que são padrões ou padrões emergentes, oriundos de organismos como OMG [21] e W3C [22]. Esta abordagem facilita o intercâmbio de conteúdos com semântica embutida entre ferramentas heterogêneas.

1.4 Organização deste texto

O restante deste trabalho está estruturado conforme descrito a seguir.

O Capítulo 2 apresenta alguns aspectos relacionados ao contexto das regras de negócio, incluindo a importância das regras nas organizações, discussões sobre conflitos entre negócios ágeis e sistemas rígidos, caracterização do problema tratado nesta tese e propostas de solução e alguns trabalhos relacionados.

O Capítulo 3 apresenta os principais fundamentos nos quais esta proposta se baseia. Incluem-se conceitos e aspectos relacionados a regras de negócio e modelagem de regras de negócio. Incluem-se também discussões sobre aspectos de gerenciamento de regras, ontologias e mecanismos de regras. Além disto, discutem-se alguns padrões e tecnologias em regras de negócio, detalhes do metamodelo SBVR, e padrões e tecnologias em processos de negócio.

O Capítulo 4 detalha a arquitetura proposta, descrevendo os módulos que a compõem e os repositórios de modelos e metadados que suportam a abordagem proposta. Apresentam-se alguns exercícios de transformação de regras descritas na terminologia das pessoas de negócio para regras computacionalmente executáveis.

O Capítulo 5 fornece uma visão geral de como foi realizada a prova de conceito para a abordagem proposta. Incluem-se o cenário usado para exercitar o uso do IDE (Integrated Development Environment) na criação e manutenção de regras de negócio e alguns aspectos da criação e edição dos elementos tratados pelo Repositório de Regras e Repositório de Ontologias. Incluem-se também alguns aspectos das etapas de preparação e de execução do serviço

No Capítulo 6 são revistas as contribuições do trabalho acrescidas de alguns comentários que ilustram as propostas de soluções para os problemas e são discutidos alguns tópicos que merecem pesquisas futuras para aprofundar o debate ou que possam derivar novas idéias.

Capítulo 2

O Contexto das Regras de Negócio

Neste capítulo, apresentam-se alguns aspectos relacionados ao contexto das regras de negócio. A seção 2.1 fornece uma visão geral sobre as regras de negócio nas organizações e na seção 2.2 discutem-se alguns conflitos existentes entre negócios ágeis e sistemas rígidos. Na seção 2.3 fala-se da importância das regras de negócio para as organizações e na seção 2.4 faz-se uma caracterização do problema tratado nesta tese e propostas de solução. Na seção 2.5 discutem-se alguns trabalhos relacionados.

2.1 Regras de Negócio nas Organizações

Qualquer negócio em qualquer organização sofre influência de um conjunto de regras que determinam como o negócio deve ser realizado. Estas regras têm suas origens em diversas áreas e assuntos de interesse da organização, tais como suas políticas, seus objetivos, suas obrigações, etc. [23]. As regras de negócio, que delimitam e governam a realização dos negócios da organização são fundamentais pois, no conjunto, representam o seu comportamento. Na realidade, qualquer mudança nas estruturas ou nos processos pode levar a uma reformulação destas regras de negócio.

Uma correta avaliação do contexto em que se insere a organização pode acarretar a necessidade de reordenação de suas políticas, objetivos e obrigações o que pode levar a mudanças nas regras de negócio. Portanto, como conseqüência desta avaliação, estabelecem-se os objetivos a serem atingidos e as estratégias para atingi-los. A partir destas definições, poderão surgir novas regras de negócio ou tornar-se necessário mudar regras existentes.

Atualmente as organizações são confrontadas com todos os tipos de regras de negócio. Instituições reguladoras, unidades empresariais, clientes, parceiros, competidores e as

condições do mercado global geram mudanças de regra a todo instante, as quais são traduzidas em regras de negócio.

O que a comunidade de pesquisa em regras de negócio procura encontrar é uma forma fácil, rápida e segura de refletir nos sistemas computacionais as mudanças das regras de negócio que ocorrem no dia a dia de qualquer organização.

Toda aplicação de uma forma ou de outra implementa algumas regras de negócio. A diferença básica entre uma aplicação e outra está na forma como as regras de negócio são criadas, implementadas e administradas. Numa aplicação as regras de negócio podem estar totalmente "misturadas" nas especificações e no código gerado e noutra podem estar estruturadas, categorizadas e devidamente relacionadas com o código gerado.

As interações entre usuários, governos e empresas sofrem freqüentes mudanças para se adaptar às novas formas de relação entre eles. Estas freqüentes mudanças nas interações resultam em mudanças nas regras de negócio. A rápida implementação destas mudanças nas regras torna-se fundamental para governos e empresas que buscam eficiência e eficácia na disponibilização de serviços e produtos aos usuários, governos e empresas. Neste sentido, é importante a criação de mecanismos que tornem o mais independente possível as regras de negócio das porções mais estáveis.

Os mecanismos de regras surgiram no início dos anos 90, comercializados pelas empresas Fair Isaac Corp., ILOG e Pegasystems Inc., entre outras. Nos últimos anos, porém, muitos fornecedores entraram no mercado e estão percebendo que com os mecanismos de regras obtém-se uma maior flexibilidade em operações de negócio, porque estes mecanismos de regras criam condições para que as aplicações e processos de negócio possam se adaptar rapidamente às demandas do mercado.

Conforme visto anteriormente, as regras de negócio representam as políticas definidas pelas organizações, na determinação de como os negócios serão realizados. Numa organização comercial, tais políticas poderiam ser aplicadas para, por exemplo, definir preços, definir estratégias de vendas, minimizar riscos, etc. Uma regra de negócio simples poderia ser expressa como: "Se valor total da compra for maior que R$300,00 então conceder 10% de desconto". Os gerentes de negócio interessam-se por esta regra na medida em que ela faz parte da política que define uma estratégia de vendas. Uma

facilidade de grande interesse dos gerentes de negócio seria poder modificar facilmente os valores R$300,00 e 10%, de modo a refletir as condições variáveis do mercado. Provavelmente, eles gostariam também de poder modificar os critérios para incluir outros testes, como número de artigos no carrinho de compras ou modificar a ação para, por exemplo, substituir o desconto por pontos no programa de fidelidade.

Numa organização existem regras de negócio formais, explicitamente definidas, tal como a obrigação de enviar um produto somente após confirmado o pagamento, mas existem também regras de negócio informais e muitas vezes tácitas ou implícitas que podem surgir em resposta a situações específicas. Embora esta característica tácita das regras de negócio seja muito comum, e também inevitável, estas regras podem trazer prejuízo para as organizações devido ao desconhecimento ou ao pouco valor dado a elas [24]. Na verdade, as regras de negócio fazem parte da memória da organização e necessitam dos mesmos cuidados que quaisquer outras informações dessa natureza.

Muitas organizações têm investido na padronização, formalização e integração de suas informações e processos. Essas atividades propiciam a oportunidade de evidenciar, reavaliar e formalizar as regras de negócio. Porém, estas tarefas não são triviais, porque a documentação das regras não é uma prática generalizada nas organizações. Contudo, imagina-se que quanto mais formalizado for um processo organizacional, melhor será a documentação de suas regras.

Dentre os processos organizacionais, os mais formais são os representados em sistemas computacionais, já que eles são definidos em linguagens formais rigorosas e, portanto, menos suscetíveis a imprecisões e ambigüidades. Logo, as regras de negócio deveriam ser identificáveis com maior facilidade e naturalidade nos sistemas computacionais. Porém, a prática não confirma este raciocínio. O problema é que as regras de negócio estão fortemente acopladas a outros aspectos da lógica do negócio e isto dificulta muito a sua identificação e individualização. Conseqüentemente, a manutenção fica muito cara e demorada.

2.2 Conflito entre Negócios Ágeis e Sistemas Rígidos

As organizações, empresas e órgãos de governo, percebem o conflito inerente entre a agilidade com que eles querem realizar seus negócios e a rigidez e dificuldade em realizar as mudanças nos sistemas que suportam os negócios. Como conseqüência, a incompatibilidade entre negócios ágeis e sistemas rígidos anula os esforços para mudar o negócio rapidamente e capitalizar em oportunidades de negócio [3]. Muitos desses conflitos têm sua origem no tratamento computacional dado às regras de negócio.

O conflito mais comum é o descompasso entre as velocidades de mudança das regras de negócio e de suas representações computacionais. Hoje em dia, os negócios têm uma dinâmica muito maior do que tinham há 30 anos. Conforme dito anteriormente, existem várias razões para os negócios mudarem. Por exemplo: lançamentos de novos produtos e serviços; parcerias, aquisições e fusões entre empresas; mudanças de legislação; novas determinações de órgãos reguladores; alterações nas condições de mercado, como atividades da concorrência ou mudanças de hábito dos clientes [25].

Com o aumento da concorrência, velocidade dos meios de comunicação e o maior grau de informação dos clientes, as organizações têm tido que fazer mudanças num ritmo muito mais acelerado que em décadas passadas. Em outras palavras, as regras de negócio têm de ser modificadas dentro de períodos de tempo relativamente curtos. Em conseqüência disso, tem-se que as soluções computacionais para os problemas dos negócios não conseguem acompanhar o ritmo de alterações a contento. Isso acontece principalmente por dois motivos [25]: 1) as regras de negócio não são separadas de aspectos de implementação dos sistemas; e 2) geralmente, os sistemas têm documentação muito precária.

De maneira geral, os sistemas legados têm documentação deficiente e, por causa disto, transformam-se repetidamente em caixas-pretas cujo conteúdo é desconhecido. Como conseqüência, quando há necessidade de alterar-se alguma regra de negócio, os engenheiros de software não têm condições de determinar precisamente quais são os impactos que essas mudanças causarão no sistema. Isso significa que existe um risco potencialmente alto de que as alterações gerem efeitos colaterais indesejados. Com o passar do tempo, após sucessivas mudanças (e muitos defeitos), é possível que os analistas de negócios comecem

a desconfiar da acurácia dos sistemas de informação e a se perguntar se o sistema está correto sob o ponto de vista do negócio.

Assim, a abordagem tradicional de desenvolvimento de software apresenta fraquezas no que diz respeito a responder rapidamente aos requisitos de negócio que mudam com muita freqüência. Pior, os requisitos podem mudar durante o desenvolvimento do sistema, ou seja, requisitos de aplicação identificados por analistas de negócio na fase inicial do projeto provavelmente vão mudar até que o sistema seja finalizado e entregue.

2.3 A Importância das Regras de Negócio

Diante do mercado altamente competitivo de hoje que exige que os negócios sejam flexíveis e admitam mudanças rápidas nos sistemas computacionais, os sistemas baseados em regras prometem contribuir para encurtar o tempo de manutenção.

Por sua própria natureza, um mecanismo de regras de negócio mantém uma parte da lógica de negócio explícita e devidamente externalizada em relação aos processos que realizam ações específicas do negócio, tais como, atualizar dados, transferir valor, autenticar usuário, etc. Em outras palavras, um mecanismo de regras permite a separação das porções que implementam as regras de negócio das porções que implementam o código procedimental. Em princípio as regras de negócio podem ser atualizadas sem necessidade de desativar as aplicações, além de facilitar o entendimento e permitir um maior envolvimento do analista de negócio no projeto e manutenção de aplicações.

Tecnicamente, as regras de negócio fazem parte de mecanismos de validação, de roteamento de mensagens, de transformação de mensagens, de sistemas de *workflow*, de atualização de banco de dados ou da lógica da aplicação, mas não deveriam estar embutidas no código executável. As regras de negócio deveriam ser externalizadas independentemente do local onde elas serão implementadas ou executadas. A externalização das regras de negócio facilita a representação de pontos de decisão para controlar e orquestrar a execução dos processos. Tipicamente, uma lógica de negócio inclui vários pontos de decisão. Estes pontos de decisão geralmente têm um efeito no fluxo de execução dos processos. Por exemplo, o índice de crédito, que mede a saúde financeira de uma pessoa em termos de

pagamentos de compras feitas a crédito, pode determinar a aprovação automática ou não de um empréstimo. Estas decisões são avaliadas tendo como base certas condições e fatos que podem ser internas ou externas ao processo de negócio e às políticas e regras da organização. Na maioria das vezes as mudanças no nível dos negócios não acarretam mudanças no nível dos processos isolados e sim, somente nos aspectos que definem os pontos de decisão [26].

Ao externalizar as regras de negócio e disponibilizar a sua definição e modificação aos analistas de negócio, o tempo entre a especificação do requisito para a mudança da regra e a respectiva implementação pode ser reduzido drasticamente. A lista a seguir mostra onde ocorre a economia de tempo: (i) o analista de negócio pode, ele próprio, mudar as regras ao invés de especificar um pedido de mudança de regra; (ii) não é mais necessário codificar em linguagem de programação convencional, na medida em que o analista de negócio já o fez na linguagem de negócio; (iii) o analista de negócio ocupará menos tempo validando as mudanças, uma vez que ele já as fez diretamente; (iv) como as regras estão externalizadas, só as mudanças nas regras precisam ser atualizadas, o que torna este procedimento mais rápido que a atualização em componentes de software com as regras embutidas.

A externalização possibilita o reuso das regras de negócio por várias aplicações, garantindo consistência em toda a organização. Isto é particularmente importante quando a organização decide oferecer os mesmos serviços em diferentes canais de comunicação, quer seja com clientes, parceiros, sócios ou uma combinação de usuários.

Outro aspecto importante nos mecanismos de regras de negócio é que eles permitem definir boa parte da lógica de negócio de maneira declarativa. Conforme sugere Date [27], há um movimento saindo do campo da programação procedimental, que enfatiza a codificação dos passos para realizar algo, para o campo da programação declarativa, que enfatiza somente a especificação do que se deseja realizar, deixando que o sistema encontre efetivamente a melhor forma de realizar o trabalho. Assim, diferentemente da programação procedimental, cujas linguagens mais conhecidas são Java, C e Fortran, que explicitamente define o algoritmo para alcançar uma meta, a programação declarativa, cujas linguagens mais conhecidas são SQL, Prolog, e WebML, explicitamente descreve somente a meta que se pretende alcançar.

2.4 Caracterização do Problema e Proposta de Solução

O ambiente de negócio muda constantemente e, como conseqüência, as exigências e as restrições nas interações entre os atores também precisam mudar para que fiquem de acordo com a nova realidade dos negócios nas organizações. A inclusão de novos produtos, a expansão em novas regiões geográficas, a expansão da rede de provedores ou a aderência a regulamentos, contratos e leis requerem mudanças na configuração das regras que regem e controlam as restrições nas interações entre o novo conjunto de atores.

Esta situação exige implementação rápida de soluções nos sistemas computacionais que suportam os negócios, para contemplar as mudanças na configuração de regras. O problema é que a forma como a maioria dos sistemas foi projetada e implementada torna a tarefa de contemplar as mudanças excessivamente demorada e cara, se não inviável, porque a razão de ser da mudança pode deixar de existir antes de se concluir a mudança. A seguir estão descritas quatro das principais razões identificadas que explicam esta dificuldade na manutenção de sistemas computacionais.

Primeiro, a dificuldade na individualização e separação das regras de negócio de outros aspectos do negócio e do sistema. Isto é decorrência do fato de que as regras de negócio estão espalhadas ou implícitas em documentos, como procedimentos, contratos, regulamentações e leis que definem as estratégias, as políticas e as relações entre as organizações. A partir destes documentos, os engenheiros de software começam a especificar os sistemas computacionais, apoiados por ambientes de modelagem e desenvolvimento de sistemas. Estes ambientes provêem ferramentas automatizadas que produzem diagramas, esquemas e modelos de software, que por sua vez facilitam o processo de geração (se possível automática) de código executável.

As regras de negócio, que antes estavam contidas nos documentos, passam a ser incorporadas ao código computacional. Isto ocorre porque atualmente o processo de desenvolvimento de software, usado nos ambientes de desenvolvimento tradicionais, não separa as regras dos outros requisitos e, portanto, não procura captar e identificar as regras de negócio, desde o início da concepção do software. Em outras palavras, nestes ambientes,

os modelos de software mostram como os dados de negócio fluem pela organização, ilustram a seqüência de ações sob determinadas condições, e mostram a estrutura da informação, sem, no entanto, isolar e expor com clareza o conjunto de regras que determinam como um negócio deve funcionar [28].

Portanto, grande parte das dificuldades no tratamento das mudanças nas regras de negócio e de soluções integradoras tem sua origem na forma monolítica e pouco estruturada da imensa maioria das aplicações computacionais, hoje existentes.

Segundo, da mesma forma que os tradicionais ambientes de modelagem e desenvolvimento de sistemas a perspectiva com que os mecanismos de composição de processos foram construídos não incorporam facilidades para individualizar e separar aspectos das regras de negócio de outros requisitos dos processos de negócio. Nesta área há muita discussão sobre a necessidade de convergência de perspectivas entre regras de negócio e processos [29, 30, 31].

Os mecanismos de composição de serviços baseados em processos, tais como o BPEL [32] ou *workflow* [33], apresentam fraquezas importantes em relação ao suporte para integração de regras de negócio [34]. O problema neles é que toda a lógica de negócio em que se baseia um processo de composição de serviços é expressa como uma especificação de processo, em um bloco monolítico. Cada restrição ou política de negócio que deve fazer parte de um processo de composição tem de ser expressa em termos de atividades, de maneira integrada com a especificação do processo.

Outro problema é a falta de flexibilidade, pois as linguagens orientadas a processo assumem que a composição é predefinida e, portanto, não facilitam a evolução. A única maneira de acomodar mudanças consiste na redefinição do processo de composição, conseqüentemente, não há nenhum suporte para, dinamicamente, realizar mudanças nos processos. Flexibilidade e adaptabilidade são características muito importantes, especialmente no contexto altamente dinâmico de composição de serviços, onde as organizações podem freqüentemente mudar as regras de negócio e as condições de colaboração.

Os mecanismos de composição de processos de negócio baseados em *workflow* [33] além de proverem suporte para interações humanas com os processos de negócio foram

muito explorados para integrar processos automatizados e semi-automatizados. Porém, o fato de não considerarem as regras de negócio de maneira isolada, quando surge uma necessidade de mudança, os sistemas mostram-se muito rígidos e fortemente acoplados e é preciso investir muito esforço para realizar a mudança. Por outro lado, os mecanismos de composição de processos mais recentes, principalmente os baseados na linguagem BPEL, contemplam fortemente os processos automatizados. Estes mecanismos, embora minimizem um pouco o esforço para realizar mudanças devido ao caráter fracamente acoplado dos processos modelados como serviços, também apresentam a fraqueza nos processos de composição (controle e orquestração) ao não considerarem as regras de negócio de maneira isolada.

Assim, embora os mecanismos de composição de processos possam prover a flexibilidade necessária para modelar a orquestração de processos e para realizar a implementação e a disponibilização ampla de processos de negócio, o não alinhamento das perspectivas desses mecanismos com as dos mecanismos de regras contribui pouco para que as regras de negócio operem de forma colaborativa com os processos de negócio.

Em terceiro lugar, os mecanismos de regras não aceitam regras descritas em linguagens amigáveis para analistas de negócio que sejam automaticamente transformáveis para código executável. O eterno desafio de eliminar ou reduzir o *gap* semântico entre a linguagem usada pelos analistas de negócio e os engenheiros de software continua esbarrando na dificuldade técnica de traduzir corretamente a semântica das regras de uma terminologia para a outra. Alguns dos mecanismos de regras disponíveis, embora aceitem regras descritas em linguagens amigáveis para analistas de negócio, não realizam a transformação automaticamente para código executável. Outros, embora realizem alguma transformação automática, não conseguem fazer a ligação com os processos de negócio que se relacionam para executar uma atividade. Portanto, os mecanismos de regras, sempre obrigam a que a transformação passe pelo entendimento entre analistas de negócio e engenheiros de software para que os últimos realizem a codificação.

Em quarto lugar, a falta de padronização nas linguagens de descrição de regras dificulta o intercâmbio de regras e a interoperação entre sistemas com suporte de diferentes mecanismos de regras. Embora alguns dos principais mecanismos de regras implementem

algoritmos de inferência padrões as linguagens de definição de regras são proprietárias, na maioria dos casos.

Neste trabalho, propõe-se um modelo para desenvolvimento e execução de serviços, baseada em regras de negócio e SOA. O modelo prevê dois ambientes: (1) ambiente de desenvolvimento rápido (RAD - Rapid Application Development) e manutenção flexível de aplicações, com apoio de repositórios de vocabulários, regras de negócio e ontologias de serviços Web; e (2) ambiente de preparação e execução de serviços, com apoio de máquinas de execução de regras de negócio e de mecanismos para a descoberta e invocação de serviços, além dos repositórios que são usados concorrentemente com o ambiente de desenvolvimento. O trabalho procura focalizar as quatro principais razões identificadas que explicam a dificuldade na manutenção de sistemas computacionais.

1. A partir de termos e fatos definidos no vocabulário de uma comunidade de negócio propõe-se definir e identificar as regras de negócio, separando-as do código executável. As regras de negócio modelam as porções mais dinâmicas (mais voláteis e sensíveis às mudanças) da lógica do negócio e as porções mais estáticas são modeladas como processos básicos que realizam ações específicas da lógica do negócio.
2. Os processos de negócio que representam orquestrações criadas através de mecanismos de composição de serviços, tal como BPEL, apresentam as mesmas características das porções estáticas, são expressas como uma especificação em um bloco monolítico e podem também ser publicados como serviços em mecanismos de registro. Dessa forma, propõe-se extrair as regras de negócio de especificações de composição de serviços, numa abordagem semelhante à usada no mecanismo de composição híbrida [34]. Baseado nisto, o modelo de desenvolvimento proposto deve aproveitar as vantagens dos mecanismos de regras, no que se refere à externalização das regras e aproveitar as vantagens dos mecanismos de composição de processos, no que se refere à padronização e independência de tecnologias.
3. Propõe-se um modelo de desenvolvimento de regras de negócio descritas na linguagem (terminologia) dos analistas de negócio que serão traduzidas para linguagens executáveis de máquinas de regras. A idéia é deixar que os analistas de

negócio façam as mudanças nas regras usando uma linguagem com termos e expressões próprios do seu jargão e estas mudanças provoquem automaticamente as devidas alterações no software, conforme sugerem DeSilva e seus colegas [35]. Propõe-se que a linguagem dos analistas de negócio seja fortemente fundamentado na especificação do metamodelo SBVR, que prevê o uso de linguagens de fácil assimilação por pessoas de negócio.

4. O modelo de desenvolvimento leva em conta as propostas de modelagem de regras de negócio, independentes de plataforma, de modo a facilitar o intercâmbio de modelos de regras entre ferramentas tecnologicamente heterogêneas, considerando a alta distributividade e complexidade das conexões entre as aplicações Web da atualidade.

2.5 Trabalhos Relacionados

Embora não tenha sido identificado na literatura nenhum outro trabalho semelhante, considerando a cooperação entre o ambiente de desenvolvimento e o ambiente de execução de regras de negócio na descoberta e execução de serviços, existem alguns trabalhos relacionados à proposta desta tese. Todos os trabalhos aqui relatados mencionam a ausência de um tratamento adequado para que os mecanismos de composição de Serviços Web trabalhem de forma colaborativa e integrada com mecanismos de regras de negócio. Como conseqüência, todos eles alegam que, as tecnologias e padrões existentes para desenvolvimento sistemas empresariais não conseguem satisfazer as exigências de realização de negócios ágeis e dinâmicos.

2.5.1 Composição Híbrida

Charfi e Mezini propõem uma abordagem de composição híbrida [34] combinando regras de negócio extraídas de especificações de composição de serviços Web, descritas, por exemplo em BPEL. Eles discutem duas alternativas tecnológicas para implementar regras de negócio em unidades modularizadas: uma usando conceitos de programação orientada a

aspectos (AOP - Aspect Oriented Programming) [36] e outra usando máquinas de regras. Defendem que a abordagem permite uma maneira mais flexível e modular para composição de Serviços Web que o uso puro e simples de mecanismos de composição de Serviços Web baseados em processos, tais como o BPEL.

A falta de flexibilidade está fortemente relacionada à falta de modularidade. Dividindo-se a lógica de negócio de uma composição em várias partes ou módulos, a composição fica mais flexível, na medida em que cada uma destas partes pode evoluir independente do resto. Isto conduz à necessidade de uma abordagem com maior granularidade. Dessa forma, com suporte apropriado, partes da lógica de composição podem ser criadas, modificadas ou apagadas dinamicamente em tempo de execução.

Para isto, os autores investigaram uma abordagem híbrida que combina regras de negócio com processos de negócio, especificados como composição de serviços Web. A idéia é que a especificação de composição só defina o controle básico e o fluxo de dados entre os serviços a serem compostos. Os aspectos da composição sensíveis às políticas e restrições, portanto sujeitas a mudanças, são modeladas como regras de negócio, modularizadas em unidades separadas. Deste modo, quando as políticas de negócio mudam, é necessário modificar somente as unidades correspondentes que modularizam a implementação das regras de negócio. Além disso, esta separação reduz a complexidade da composição e aumenta a adaptabilidade. Abstraindo-se da fase de análise que teria como resultado a identificação das regras de negócio e partindo-se para a fase de implementação onde regras são modularizadas e implementadas com alguma tecnologia específica os autores propõem duas alternativas.

A primeira alternativa baseia-se no conceito de AOP para modularizar a implementação das regras de negócio. A implementação das regras de negócio tende a ser transversal a várias atividades de um processo. A AOP provê meios para modularizar as regras de negócio, minimizando os efeitos desta característica transversal, da mesma forma que já se mostrou útil para modularizar regras de negócio no contexto da orientação a objetos [37]. Nesta alternativa mapeiam-se os conceitos de regras de negócio nos conceitos de AOP e delineia-se como as regras de negócio podem ser implementadas na proposta de extensão de BPEL orientada aspectos, chamada de AO4BPEL [38].

Uma grande fraqueza nesta alternativa é que obriga o programador a implementar as regras usando aspectos em AO4BPEL e a configurar as interações das regras com o processo. Além disso, o programador precisa também administrar todas as regras, incluindo verificação de dependências, verificação de consistência, combinação de regras e resolução de conflitos. Conforme afirmação dos próprios autores, a configuração manual de combinações de regras é um grande incômodo. Além disto, as regras de negócio são especialmente difíceis de implementar como aspectos porque eles demandam raciocínio lógico e combinação de regras.

A segunda alternativa para modularizar a implementação de regras de negócio combina a especificação do processo de composição com sistemas dedicados para especificação declarativa de regras de negócio, tais como Jess [39]. A Figura 2.1 ilustra a arquitetura desta alternativa.

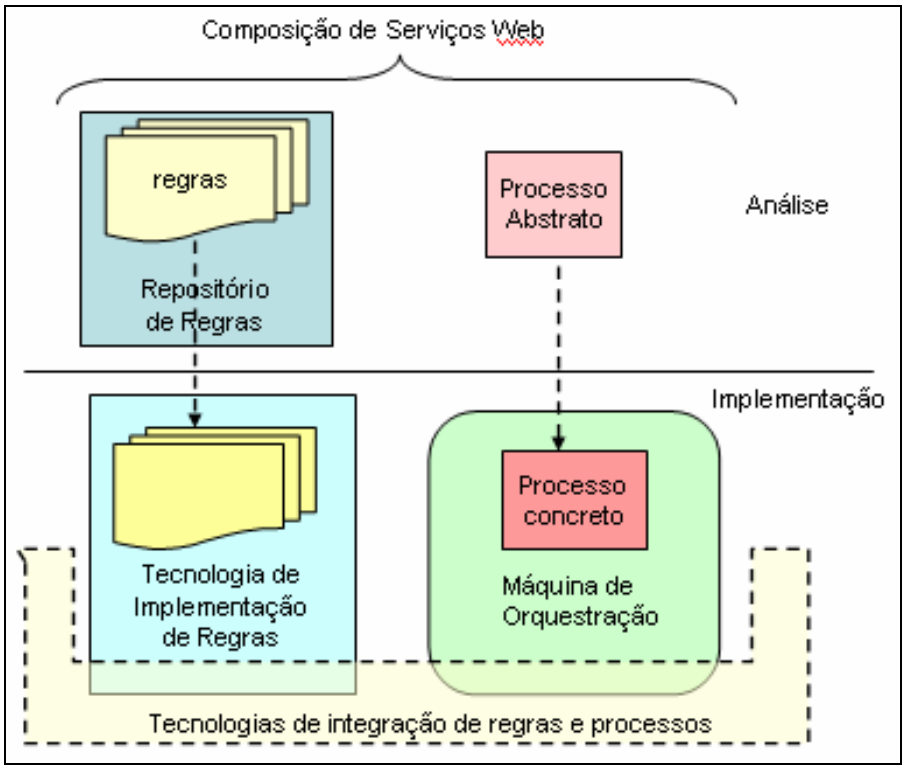

Figura 2.1 - Arquitetura da Composição Híbrida.

Nesta alternativa propõe-se o uso de um mecanismo de integração de regras e processos ao invés do uso de um mecanismo de regras "puro", no qual toda a lógica de composição é

especificada na forma de regras de negócio. Segundo os autores, o uso de mecanismos de regras "puros" não é apropriado porque não haveria uma visão global da composição. Considerou-se as regras de negócio como parte da implementação de composição de Serviços Web, ou seja, elas são comparáveis aos processos de negócio executáveis em BPEL. No entanto, diferentemente dos processos de negócio abstratos em BPEL que especificam os protocolos de negócio, as regras nesta alternativa não são publicadas entre os parceiros. Portanto, não são visíveis para os usuários do serviço Web composto resultante. O que enfraquece esta alternativa é que ela apresenta uma grande questão em aberto. Esta questão reside exatamente no aspecto mais importante da proposta que é o que trata da integração de regras e processos. Os autores alertam para o fato de que a simplicidade do ponto de vista do usuário é acompanhada por uma complexidade maior na máquina de orquestração subjacente. A camada de integração, mostrada na parte inferior da Figura 2.1, torna-se mais complexa para realizar a execução dos processos porque deve ser integrada com a implementação das regras de negócio.

2.5.2 *Framework* BCDF

Bart Orriens e Jian Yang propuseram um *framework*, que chamaram de BCDF (*Business Collaboration Development Framework*) [40] para aumentar a flexibilidade e adaptabilidade de sistemas computacionais, onde regras de negócio são usadas para dirigir e restringir colaborações entre processos de negócio. Argumentam que a abordagem aumenta a flexibilidade devido ao fato de que o desenvolvimento de colaborações é conduzido por regras de negócio, que podem ser encadeadas e usadas para automatizar diagnósticos e tomadas de decisão complexas. A adaptabilidade pode ser aumentada porque as mudanças podem ser realizadas com um mínimo de interferência nas colaborações existentes.

A idéia do BCDF apresenta um certa similaridade com a proposta de composição híbrida [34] na medida em que também se baseia na provisão de facilidades para separar aspectos das colaborações de negócio e modularizá-las como regras de negócio. A Figura 2.2 fornece uma visão geral, em três dimensões, dos conceitos envolvidos nesta proposta.

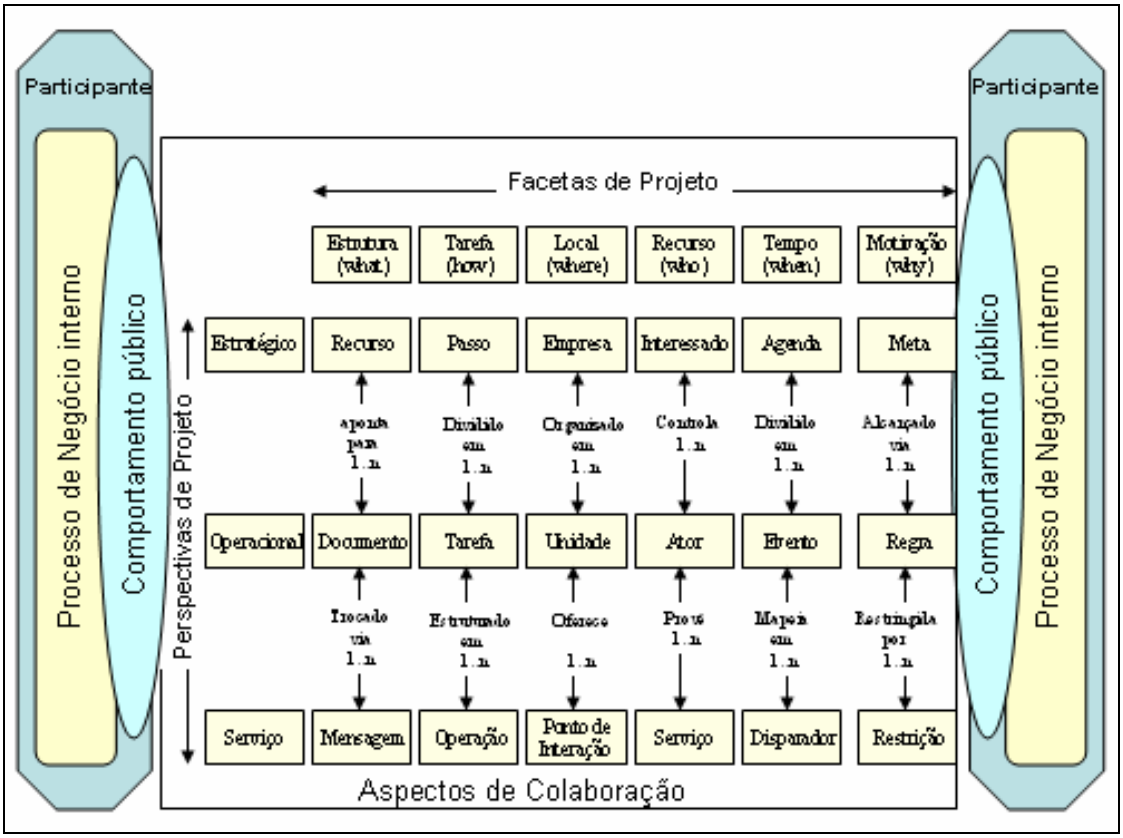

Figura 2.2 - Business Collaboration Development Framework.

A Primeira dimensão refere-se a aspectos de colaboração enfatizando os diferentes comportamentos de uma organização em relação às colaborações entre seus processos de negócio. Nesta dimensão o objetivo do desenvolvimento preconiza a captura do comportamento privado (*private behavior*) nos aspectos dos processos de negócio internos (*internal business process aspect*) e baseada neste comportamento privado a organização pode especificar suas capacidades em termos do seu comportamento público (*participant public behavior aspect*). Subseqüentemente, a organização pode negociar com parceiros para estabelecer uma colaboração.

A segunda dimensão identifica três níveis de abstração diferentes para separar os aspectos ao tratar de requisitos de negócio, requisitos técnicos e dependências entre eles, ou seja, todos os aspectos importantes quando as organizações tentam cooperar entre si. O nível estratégico, no qual as organizações descrevem o propósito e os requisitos em alto

nível para uma colaboração entre processos de negócio. O nível operacional, no qual descrevem-se as condições operacionais sob as quais as organizações podem cooperar e onde se define como os objetivos serão realizados em termos de atividades empresariais concretas. O nível de serviço, no qual define-se a realização técnica das atividades empresariais e onde descrevem-se as interações entre os serviços dos diferentes parceiros na colaboração.

A cada nível de abstração é necessário considerar vários aspectos, tais como, planejamento, uso de recursos e otimização logística no nível estratégico. Para reduzir a complexidade no tratamento de todos estes aspectos, a terceira dimensão baseia-se em facetas (*facets*) para modularizar as definições de colaboração entre processos de negócio. Isto ajuda as organizações na descrição de diferentes contextos a partir dos diferentes níveis em que uma colaboração entre processos de negócio pode ser observada. As facetas identificadas são: (i) faceta *what* que enfatiza a visão estrutural, (ii) faceta *how* enfatizando o ponto de vista funcional, (iii) faceta *where* que expressa a localização geográfica, (iv) faceta *who* que enfatiza os participantes na colaboração, (v) faceta *when* que trata do aspecto temporal, e (vi) faceta *why* que trata da razão. As facetas provêem uma cobertura completa para cada nível individual, onde a semântica depende do nível que elas modularizam. As interações entre facetas refletem as relações que existem entre diferentes contextos, tal como a interação entre o contexto temporal de uma colaboração e seu fluxo de controle.

Para discutir aspectos de colaboração, níveis e facetas os autores usam dois tipos de modelo: metamodelo e modelos definidos para níveis individuais. Metamodelos provêem diretrizes de projeto em termos de classes e das relações entre elas. Dependendo do aspecto de colaboração sendo modelado, restrições adicionais são definidas no metamodelo. Um modelo representa um projeto de aplicação particular e é resultante da definição de classes em um metamodelo. Todo metamodelo consiste de seis classes onde cada classe captura uma faceta em particular; ou seja, *what, how*, *where, who, when* e *why.*

Toda classe constitui-se de um conjunto de atributos logicamente relacionados. Associações conectam as classes que expressam dependências entre facetas. Mapeamentos (*mapping*) definem dependências entre níveis provendo ligações entre classes que

descrevem a mesma faceta em diferentes perspectivas (ilustrada pelas setas entre facetas de diferentes perspectivas na Figura 2.2). Dependências entre aspectos de colaboração são expressas por conexões (*connections*) que ligam as mesmas classes no metamodelo, na medida em que elas são aplicadas em diferentes aspectos.

Esta proposta, embora ainda não tenha uma implementação completa e apresente algumas questões em aberto, tais como o tratamento de qualidade de serviço, tarifação e detalhes de segurança, aparentemente dispõe de um modelo de desenvolvimento bastante sofisticado para aumentar a flexibilidade e adaptabilidade de sistemas computacionais. Um aspecto que merece melhorias refere-se à sua pouca amigabilidade na definição de regras, que devem ser descritas em RuleML [41], o que está muito distante da terminologia usada pelos analistas de negócio. Isto significa que, nesta proposta, a redução ou eliminação do *gap* semântico entre as linguagens das pessoas de negócio e dos engenheiros de software não foi levada em consideração.

2.5.3 ILOG JRules 6

ILOG JRules 6 [42] é considerado um dos mais completos sistemas gerenciadores de regras de negócio (BRMS – Business Rules Management System) do mercado. A Figura 2.3 mostra uma arquitetura integrando o uso do BRMS para executar aplicações baseadas em regras de negócio.

Com JRules 6, pode-se usar o modelo de objetos completo e XML Schemas. Objetos Java e XML são combinados no modelo de objetos de negócio (BOM - Business Object Model). A partir de modelos BOM, os objetos e seus métodos são traduzidos em fragmentos de linguagem natural. Assim, por exemplo, o objeto *Cliente* pode ser associado ao fragmento "cliente" e o método *obterIdade* pode ser associado a "a idade de". Abstraindo-se estes elementos em linguagem natural a seguinte regra pode ser construída: "**Se** a idade de cliente **for maior que** *18* **E** ... **Então** ...", onde, **texto em negrito** é usado para mostrar elementos do Rule Builders, texto sublinhado é usado para mostrar a tradução de objetos BOM, e *texto em itálico* é usado para mostrar entradas do usuário para o Rule Builder.

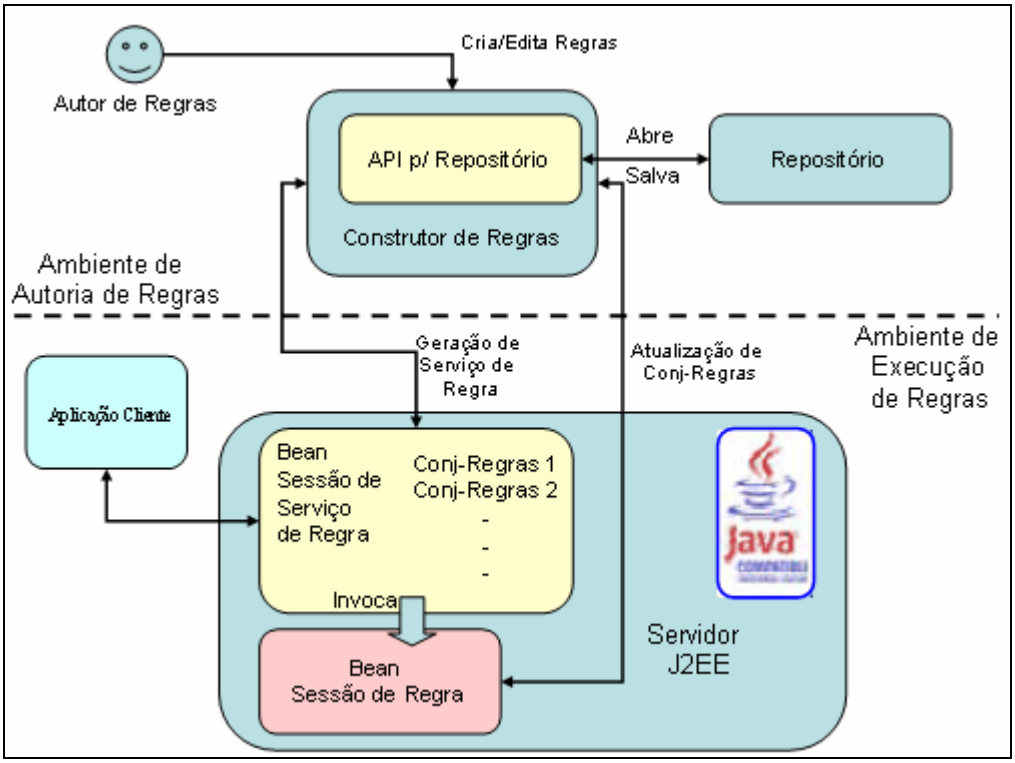

Figura 2.3 - Arquitetura do JRules BRMS.

Estas regras são organizadas em projetos de conjunto de regras (*rule sets*). Após a carga dos objetos (Java e XML) na memória de trabalho os conjuntos de regras podem ser executados tendo como base os objetos contidos na memória. As regras podem então manipular ou criar qualquer objeto. Os conjuntos de regras e os objetos BOM são gravados no repositório de regras. Portanto, os gerentes e administradores podem acessar, atualizar e substituir regras de acordo com o seu papel para cobrir todo o ciclo de vida de regras de negócio, ou seja, criar, substituir, atualizar e excluir. A Figura 2.4 mostra a interface do Rule Team Server, copiado de [43], usado por analistas de negócio para definir regras na sua terminologia, conforme pode ser visto no destaque da parte inferior da figura.

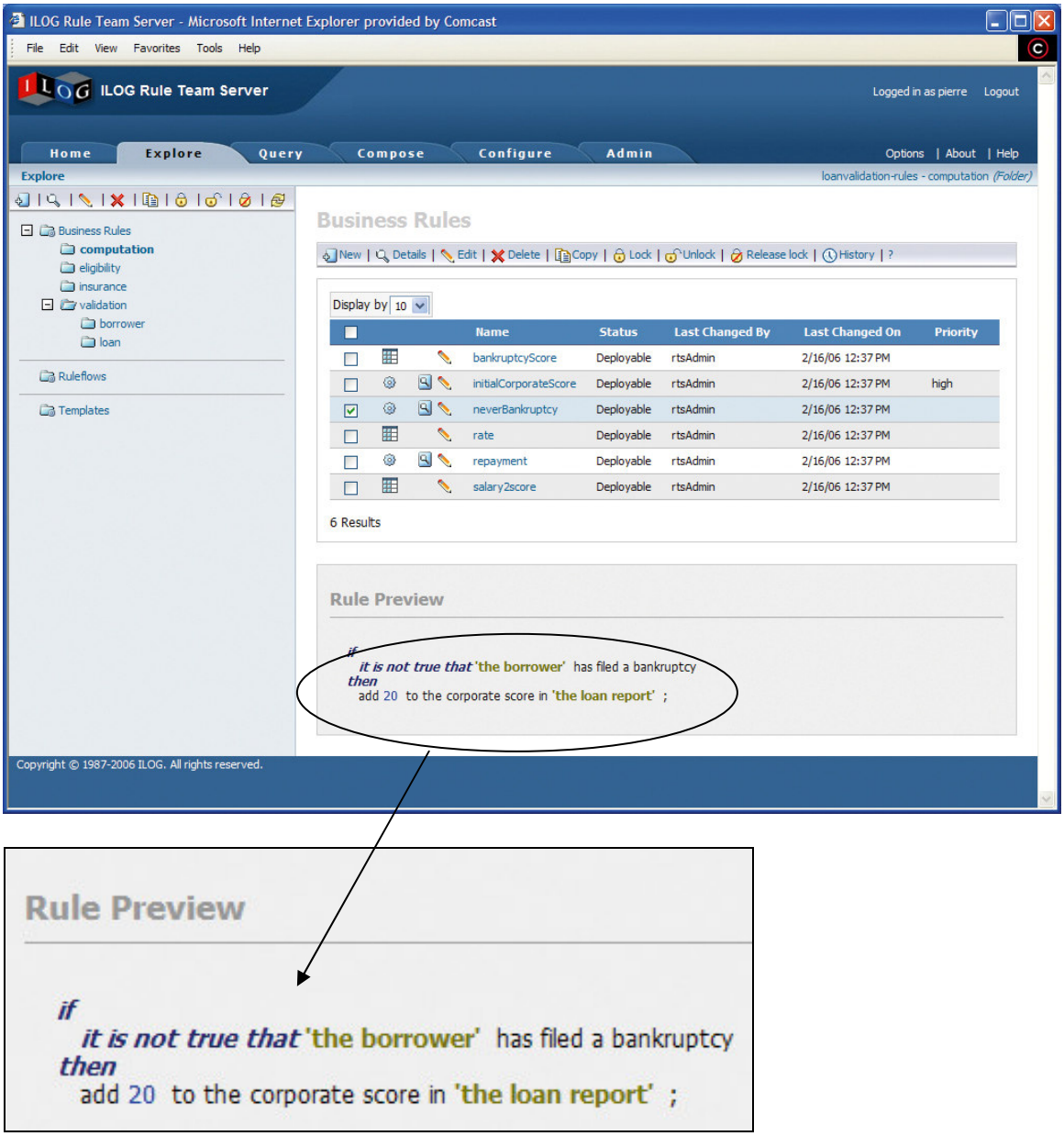

Figura 2.4 - Interface do ILOG Rule Team Server [43].

Este produto tem como vantagens estar numa versão já bem estabilizada, estar baseado no padrão JSR 94 [44] e aceitar regras descritas em linguagem natural. A desvantagem é que, embora as regras sejam descritas em linguagem natural e os métodos associados aos fragmentos possam ser operações de Serviços Web a associação precisa ser realizada por engenheiros de software. Isto ocorre porque os analistas de negócio não têm conhecimentos suficientes para realizar as associações de termos e predicados com os respectivos objetos e

métodos. Portanto, também nesse produto, não é ainda possível a redução ou eliminação do *gap* semântico entre as linguagens das pessoas de negócio e dos engenheiros de software.

2.6 Considerações Finais do Capítulo 2

Nesse Capítulo foram apresentados alguns aspectos relacionados ao contexto das regras de negócio, incluindo uma visão geral sobre as regras de negócio nas organizações e sua importância, discussões sobre conflitos entre negócios ágeis e sistemas rígidos, caracterização do problema tratado nesta tese e propostas de solução e alguns trabalhos relacionados.

Do exposto nesse Capítulo, pode-se concluir que as regras são sempre relevantes para a realização do negócio qualquer que seja o negócio e independentemente do modelo de administração, das pessoas envolvidas e até mesmo da tecnologia utilizada.

Considera-se que toda organização sofre influência de um conjunto de regras que determinam como o seu negócio deve ser realizado. Tais regras, por restringir ou provocar a ocorrência de eventos e fatos, refletem a forma como a organização se comporta e toma as suas decisões de negócio. Para reagir prontamente às mudanças das condições, a organização deve decidir que ação tomar e implementar essas decisões com a maior brevidade possível. Porém, sabe-se que a implementação das mudanças nos sistemas computacionais é imensamente mais complexa e demorada que a tomada de decisões de negócio numa organização. Isto é conseqüência da dificuldade na captura das regras de negócio, a partir de documentos e experiências das pessoas de negócio e da falta de mecanismos que consigam gerenciar as regras de negócio como entidades independentes de outros aspectos dos sistemas de informação. .

Em grandes aplicações de negócio, principalmente em sistemas legados, é bastante comum que as regras de negócio estejam “espalhadas” na lógica de negócio como um todo. Assim, efetuar mudanças controladas nestas regras de negócio é uma tarefa árdua, principalmente se considerarmos que agora, mais que nunca, as alterações precisam ser

feitas por analistas de negócio que tipicamente não têm experiência em programação convencional.

Assim, considera-se que o tratamento do problema caracterizado nesta tese passa por encontrar uma forma de refletir rapidamente nos sistemas computacionais as mudanças nas regras de negócio que ocorrem no dia a dia de qualquer organização.

No próximo Capítulo serão apresentados os principais fundamentos, padrões e tecnologias nos quais esta tese se baseia para abordar o problema citado.

Capítulo 3

Fundamentos, Padrões e Tecnologias

Uma resposta adequada à demanda por mudanças rápidas exigidas pelo mercado altamente competitivo dos dias atuais, pressupõe a combinação de várias abordagens, tecnologias e ferramentas para realizar a integração de sistemas computacionais. Neste Capítulo apresentam-se os principais fundamentos nos quais esta proposta se baseia. Assim, na seção 3.1 apresentam-se conceitos e aspectos gerais relacionados com regras de negócio. Na seção 3.2 discutem-se alguns aspectos em torno de modelagem de regras de negócio e na seção 3.3 sobre aspectos de gerenciamento de regras de negócio e ontologias e sobre aspectos dos mecanismos de regras de negócio. A análise desses aspectos é útil na elaboração e formalização de regras de negócio e na concepção de ambientes integrados de modelagem de regras de negócio. Na seção 3.4 discutem-se alguns padrões e tecnologias em regras de negócio e na seção 3.5 alguns detalhes do metamodelo SBVR. Na seção 3.6 discutem-se alguns padrões e tecnologias em processos de negócio.

3.1 Conceitos em Regras de Negócio

Conforme Houaiss [45], a palavra **regra** significa (1) aquilo que regula, dirige, rege; princípio, norma, preceito; (2) norma, fórmula que indica o modo apropriado de falar, pensar, agir em determinados casos; (3) aquilo que foi determinado, ou se tem como obrigatório, pela força da lei, dos costumes etc.; lei, princípio, norma; (4) conjunto de princípios, de normas, que perfazem os estatutos de uma ordem religiosa; prescrição

religiosa; (5) qualidade de moderado, de metódico; (6) cada uma das linhas que compõem o papel pautado; (7) Estatística: pouco usado; (8) aquilo que pode servir de modelo; exemplo. Etimologicamente, a palavra *regra* vem do latim *regùla*, que significa: (i) barra de pedreiro ou carpinteiro para aferir e tornar reta uma superfície, (ii) pau ou ripa que sustenta a alguma coisa, (iii) tala que endireita osso quebrado, e (iv) preceitos ou normas que servem de guia a procedimentos ou comportamentos. O vocábulo *regùla*, por sua vez, derivou-se do verbo *regère*, que significa: dirigir, guiar, conduzir, governar. No Português, *regùla* deu origem a *regra*, *régua* e *relha*.

Eileen Pfeiffer Flores [46] faz uma interessante análise sobre uma proposta de *Ampliação do conceito de regra.* Nesta análise, Flores, citando o autor da proposta, argumenta que

> ”Skinner adota o termo *regra* para se referir a casos que normalmente chamamos de *regra* no dia-a-dia, mas também inclui numerosos casos que vão além desse uso cotidiano. Assim, ele lista no tópico “Alguns tipos de regras” (pp.162-166), fenômenos muito diversos, como leis científicas, máximas e provérbios, resoluções e planos, modelos a serem imitados, instruções, padrões a serem seguidos (por exemplo, um padrão para bordado) e textos (ele considera qualquer texto como sendo, ele próprio, uma regra de como deve ser lido). Em outras partes do mesmo texto, cita a gramática, as normas religiosas, as normas éticas e leis governamentais, conselhos, avisos, comandos ou ordens, mapas, pedras deixadas como marcas no caminho, relógios, entre outros. Os exemplos são citados para mostrar quão amplo é o conceito de regra proposto por Skinner e como ele se constitui numa extensão do uso cotidiano...”.

Embora a proposta de ampliação do conceito de regra seja interessante, ela é contrária aos interesses desta tese, que busca e necessita, exatamente de um conceito bastante limitado aos domínios dos negócios e da computação. Assim, a redução da abrangência do conceito de regra no contexto desta tese se aproxima bastante de alguns dos usos propostos por Houaiss, quais sejam, (1) aquilo que regula, dirige, rege; princípio, norma, preceito; e (3) aquilo que foi determinado, ou se tem como obrigatório, pela força da lei, dos costumes

etc.; e da derivação etimológica (iv) preceitos ou normas que servem de guia a procedimentos ou comportamentos. Aproxima-se também de "as instruções, as normas éticas e leis governamentais, conselhos, avisos, comandos ou ordens" citadas por Flores.

3.1.1 Definições de Regra de Negócio

Os ambientes organizacionais incorporam uma grande variedade de regras. Há regras para o comportamento dos negócios, das pessoas e das próprias organizações [47]. Percebe-se que algumas regras de negócio, embora muito importantes, são puramente orientadas ao comportamento pessoal ou organizacional e, portanto, nunca serão automatizadas computacionalmente. Por exemplo, "Todo funcionário deve portar o crachá de identificação ao circular nas dependências da empresa." é uma regra de negócio, mas que dificilmente será tratada por sistemas computacionais. Assim, é necessário reconhecer que o termo "regra de negócio" é muito amplo e que existem regras de negócio que nunca serão consideradas por sistemas computacionais.

A tecnologia de regras de negócio é importante para modelar as regras como entidades independentes da aplicação, que cuidam da forma como o negócio será realizado. Embora a tecnologia de regras de negócio seja usada há vários anos, não há ainda um consenso sobre a definição do termo "regra de negócio".

Conforme o *Business Rules Group* define em *GUIDE* [48] e o *Business Rules Team* reafirma em [19] há duas definições para regras de negócio: uma que representa a perspectiva empresarial e outra que representa a perspectiva de TI. Da perspectiva empresarial, uma regra de negócio é uma premissa, que influencia ou guia o comportamento empresarial, em defesa de uma política empresarial que foi formulada com respeito a uma oportunidade, uma ameaça, uma força, ou uma fraqueza. Da perspectiva de TI, uma regra de negócio é uma declaração que define ou restringe um aspecto do negócio, ou em outras palavras, define a estrutura do negócio e controla e influencia o comportamento do negócio.

Conforme a IBM [49], embora alguns aspectos de sua proposta - especialmente sobre a origem de regras de negócio - tenham sido inspirados em *GUIDE do Business Rule Group*

[48], há uma diferença importante entre regras de negócio e os vocabulários do negócio (estrutura do negócio). Para a IBM as regras de negócio não tratam da estrutura do negócio, cuja responsabilidade é do vocabulário do negócio que define a estrutura e as relações entre as condições usadas por pessoas de negócio. Afirma ainda que regras de negócio são atômicas e contêm declarações altamente estruturadas que restringem algum aspecto do negócio, controlam ou influenciam seu comportamento. Elas são expressas usando condições definidas no vocabulário do negócio.

No contexto deste trabalho, uma regra de negócio é "**uma regra que pode ser interpretada por computadores, que define ou restringe alguns aspectos de um negócio, introduzindo obrigações ou necessidades, conforme as políticas da organização**".

3.1.2 Exemplos de Regras de Negócio

A seguir apresentam-se alguns exemplos de regras de negócio que permeiam os ambientes organizacionais. Deve ser esclarecido que a grande maioria de tais regras de negócio não está devidamente explicitada e, portanto, não está sob controle de mecanismos de gerenciamentos de regras de negócio. A falta de mecanismos eficientes para gerenciamento de regras de negócio é a causa de muitos dos problemas na inflexibilidade ao implementar as mudanças nos sistemas computacionais.

Algumas regras de negócio no contexto acadêmico:

- Um aluno **não deve** ser aceito se a classe estiver lotada.
- Um professor **deve** entregar as notas em até 3 dias após a aplicação da prova final.
- O número de orientados de um professor é resultado da aplicação da fórmula X.

Algumas regras de negócio no contexto de locação de carros:

- Um carro **deve** ter um número de registro.
- Um carro **não deve** ser entregue ao cliente se não foi apresentado o cartão de crédito como garantia de pagamento.

- Se o modelo de carro reservado não estiver disponível no instante da entrega, então um modelo superior **deve** ser disponibilizado ao cliente.

Algumas regras de negócio no contexto de identificação civil:

- Um usuário que perdeu a carteira de identidade **precisa** registrar um Boletim de Ocorrência notificando a perda.
- Uma carteira de identidade **deve** ser cancelada se um Boletim de Ocorrência foi registrado por perda.
- A carteira de identidade **não deve** ser cancelada se os dados fornecidos no Boletim de Ocorrência não forem iguais aos da carteira de identidade.

Algumas regras de negócio no contexto de passagens aéreas:

- O preço de uma passagem aérea de Campinas a Brasília é R$700,00 se a ida e a volta ocorrerem no meio da semana. E R$300,00 se a ida ou a volta ocorrerem no final de semana.
- Toda reserva de passagem perderá a validade se não for quitada com 48 horas de antecedência do horário do vôo.

Algumas regras de negócio no contexto de varejo comercial:

- Em qualquer compra, o número de unidades de artigos em promoção **não deve** ser maior que 3.
- Às terças-feiras os preços de todos os produtos horti-fruti-granjeiros terão 50% de desconto.

3.1.3 Tipos de Regras de Negócio

As regras de negócio podem ser categorizadas de várias formas, entre as quais as apresentadas a seguir parecem ser interessantes.

A. Halle [25] e Wagner [50] fazem uma boa classificação:

- Regra de Reação ou Regra de Ação - Especifica uma regra "evento-condição-ação" ou regra de produção (se-então operação). Define um conceito de negócio, ou relaciona conceitos de negócio, que em muitos casos são definidas a partir do que seriam as pos-condições de casos de usos.

- Regra de Derivação - Possibilita a descoberta de nova informação a partir de outras informações já conhecidas. Pode ser uma fórmula matemática, ou simplesmente um fato derivado da forma "se A é verdade, então pode-se dizer que B é verdade."
- Regra de Restrição - Obriga ou proíbe uma ação. Pode estar refletida em restrições de integridade de banco de dados, em regras de validação de dados, gatilhos, procedimentos, etc.

Muitos dos mecanismos de regras de negócio existentes não suportam todos estes diferentes tipos de regra descritos acima. O suporte para os diferentes tipos de regra depende exclusivamente da máquina de regras. A maioria dos mecanismos de regras suportam só as regras de reação, ou seja, na forma de regras "se-então". As regras de derivação e as regras de restrição têm de ser modeladas adequadamente na forma de regras de reação.

B. A OMG define tipos de regras de negócio de maneira bastante consistente com a lógica modal, conforme recomendação do metamodelo SBVR [19]. O metamodelo SBVR considera que para se ter uma boa interpretação das regras de negócio deve-se focalizar nos aspectos de obrigação e de necessidade das regras. Portanto, em SBVR, uma regra é "um elemento de orientação que introduz uma obrigação ou uma necessidade". Assim, para a OMG as duas categorias fundamentais de regra são:

   - Regras Estruturais (introduz necessidades) - Estas regras especificam como o negócio organiza (sua estrutura) os artefatos com os quais ele opera. Regras Estruturais suplementam as definições. Por exemplo, no contexto de locação de automóveis, uma Regra Estrutural poderia ser formulada assim:

     (Necessidade) Um cliente tem pelo menos:

     - • uma reserva de aluguel.
     - • um aluguel em progresso.
     - • um aluguel completado nos últimos 5 anos.

   - Regras Operativas (introduz obrigações ou proibições) - Estas regras governam a conduta da atividade empresarial. Em contraste com as Regras Estruturais as Regras Operativas podem ser violadas diretamente por pessoas envolvidas nos

negócios. Por exemplo, no contexto de locação de automóveis, uma Regra Operativa poderia ser formulada assim:

(Proibição) A um cliente que aparentemente esteja intoxicado ou drogado não se deve dar a posse de um carro de aluguel.

3.1.4 Diferença entre Regras de Negócio e Regras de *Workflow*

As regras de negócios têm propósitos diferentes das regras de *workflow*. As regras de negócio governam uma empresa, sem considerar o processo ou procedimento escolhido para executá-las e, como tal, elas se aplicam a qualquer projeto que a empresa realize. As regras de *workflow*, ao contrário, definem como um trabalho é realizado. A meta de muitos projetos é criar ou melhorar *workflows*, ou seja, mudar as regras de *workflow*.

As regras de negócio restringem algum aspecto da organização enquanto que as regras de *workflow* definem o processo pelo qual um negócio é realizado. Assim, as regras de *workflow* são limitadas, restringidas pelas regras de negócio.

Por exemplo, considerando-se que um banco tenha como regra de negócio, extraída de sua política de segurança e garantia, a determinação de que quando um cliente quiser sacar mais de R$ 5000 de uma conta, o caixa precisa obter autorização de um superior hierárquico. A regra de *workflow* para suportar esta regra de negócio descreve como o caixa obtém a autorização. Neste exemplo, a regra de negócio e a regra de *workflow* poderiam ser descritas como segue:

- **Regra de negócio**: Qualquer saque em dinheiro de R$ 5000 ou mais deve ser autorizado por um supervisor.
- **Regra de *workflow***: Ao processar um saque de R$ 5000 ou mais, o caixa deve preencher um formulário de aprovação e obter a assinatura do supervisor.

A mesma regra de negócio acima poderia ter suporte, pelo mesmo banco em outro instante ou por outro banco, pelas seguintes regras de *workflow*:

- Ao processar um saque de R$ 5000 ou mais, o caixa deve enviar uma mensagem, via *chat,* ao supervisor que retorna uma mensagem padrão para indicar a aprovação.

- Ao processar um saque de R$ 5000 ou mais, o caixa deve enviar um mensagem, via rede interna, ao supervisor que pressionara um campo na tela para sinalizar a aprovação.

3.2 Modelagem de Regras de Negócio

Alguns autores sugerem que as atividades de desenvolvimento de regras de negócio são semelhantes, se não idênticas, às atividades de desenvolvimento de software, conforme recomenda a comunidade de Engenharia de Software. Se não em todas as fases de desenvolvimento, pelo menos na fase de levantamento e análise de requisitos, realmente as boas práticas da Engenharia de Software podem ser aplicadas na modelagem de regras de negócio [51], conforme sugerem os seguintes passos em cada uma destas fases.

3.2.1 Levantamento e Análise de Requisitos

- Identificação de atores, contexto, objetivos e objetos a serem entregues;
- Identificação de processos de negócio
- Decomposição de processos em atividades
- Decomposição de atividades em operações do sistema
- Para todas as atividades definir controle de seqüência
- Produção de diagramas de dependência de atividades
- Para todas as atividades identificar entradas e saídas de dados
- Produção de modelos de objetos de dados

O detalhamento destas atividades, por exemplo, para a "Identificação de atores, contexto, objetivos e objetos a serem entregues" poderia ser na forma de descrições como se seguem:

- Papéis dos atores, tais como clientes, parceiros, reguladores, executivos, gerentes, operadores, etc.

- Contexto dos atores, tais como “Depto de Contabilidade trata com débitos e créditos”, “Depto de Pedidos trata com compras e vendas”, etc.(comunidade)
- Objetivos do negócio, tais como minimizar os pagamentos atrasados, produzir pedidos de compras livres de erros, etc. (políticas)
- Objetos a serem entregues, tais como contas de novos clientes, documento de pedido de compra, etc. (artefatos)
- Identificação de processos de negócio (casos de uso)
- Ações em alto nível, na terminologia do negócio
- Identificação dos objetos desejados e as respectivas entradas necessárias

3.2.2 Detalhamento de Requisitos

Esta abordagem de detalhamento de requisitos preconiza que a cada requisito deve ser feito um detalhamento sucessivo em vários níveis, nas restrições e definições de termos, onde cada termo é uma regra. Ou seja, considerando-se que as regras de negócio são asserções sobre dados pode-se dizer que estas definições de termos são as regras de negócio. Tomando-se um exemplo de aplicação financeira, pode-se dizer que “o saldo de um cliente é a soma de todos os pedidos em aberto, ou seja, não pagos“. Esta definição não é ambígua e certamente tem impactos em várias transações relacionadas, tais como inserir ou excluir pedidos.

Considere o seguinte exemplo no contexto de um Pedido de compra:

1. Começando-se com a exigência “checar o Limite de crédito“ a primeira regra praticamente só redeclara a exigência. Agora pode-se perguntar “o que é Saldo?”.
2. A resposta para esta pergunta resulta na segunda regra: o Saldo é a soma dos Totais dos Pedidos não pagos. Isso conduz à pergunta: o que é Totais dos Pedidos?
3. A resposta para esta pergunta é que Totais dos Pedidos é a soma de Frete + Total dos itens. Assumindo-se que o Frete seja fornecido, pode-se derivá-lo de regras

relacionadas sem nenhum impacto. Assim, é necessário perguntar: o que é Valor de item?

Desta forma, identificando-se termos e regras sucessivamente a partir de necessidades iniciais que são expressas por frases curtas e simples.

As sugestões e idéias aqui discutidas podem ser aproveitadas na concepção de um ambiente de desenvolvimento (IDE – Integrated Development Environment) de regras de negócio. Este IDE poderia fazer uso de repositórios de termos, conceitos e regras na forma de vocabulários ou ontologias específicos de domínio de aplicação ou negócio.

O IDE proposto no capítulo 4 foi baseado nestas idéias e pressupõe o uso de um repositório de vocabulário e regras de negócio e uma ontologia para estruturar os serviços Web.

3.3 Gerência de Conhecimento e Mecanismos de Regras de Negócio

Iniciativas bem sucedidas em diálogos orientados por bases de conhecimento podem ser encontradas em centrais de atendimento e *help desks*. Novas aplicações com alto potencial para se beneficiar desses sistemas são as aplicações baseadas na Web e de auto-atendimento. Estas aplicações devem ter diálogos bem estruturados, pressupondo ambientes onde a intervenção humana precisa ser mínima e tudo está sujeito a mudanças rápidas **[52]**. Portanto, tais diálogos caracterizam-se por

1. apresentar a questão correta no instante correto;
2. apresentar sugestões e heurísticas automaticamente em pontos estratégicos;
3. eliminar opções, alternativas ou conclusões desnecessárias; e
4. inserir dinamicamente questões oportunistas, tais como para "venda casada".

A necessidade de se transferir e codificar conhecimentos entre diferentes sistemas e grupos de usuários conduz à necessidade de estruturar-se o conhecimento e, isto pressupõe a definição de regras de negócio. Portanto, há uma grande oportunidade para fazer conexão

entre regras de negócio e gerência de conhecimento, tratando as características listadas através de regras de negócio.

3.3.1 Estrutura de Diálogos orientada por Regras de Negócio

Um infra-estrutura de portal Web, tal como o proposto na prova de conceito deste trabalho, também apresenta características dos diálogos citados anteriormente. Por ser responsável pela captura dos desejos do usuário, a infra-estrutura precisa prover interações bem estruturadas com o intuito de reduzir ao mínimo a intervenção humana.

Por exemplo, quando o usuário, interagindo com a infra-estrutura, comunica um evento de vida, tal como "Perdi minha carteira de identidade" a infra-estrutura poderá incluir a questão "Deseja solicitar a Segunda Via da sua Identidade?". Esta questão apresenta pelo menos duas ou três das características listadas, quais sejam, ela elimina as questões alternativas "Deseja registrar um BO?" e "Deseja cancelar a Identidade?" uma vez que estas, naquele instante e contexto, são desnecessárias, pois a "solicitação da Segunda Via da Identidade" inclui necessariamente o "Registro do BO" e o "Cancelamento da Identidade" perdida; ela é a questão correta e oportunista inserida no instante correto, pois quaisquer das questões alternativas acarretariam perda de tempo e poderiam realizar um serviço incompleto do ponto de vista do negócio.

Outro exemplo ocorre quando o usuário comunica o desejo "Quero abrir uma loja de ..." a infra-estrutura poderá incluir uma questão genérica como "Deseja montar processo de abertura de firma?" que incluiria "solicitação de certidões negativas" (nas esferas municipal, estadual e federal), "solicitação de licenças" (ambientais, sanitárias, etc.), pagamento de taxas, etc.

Num terceiro exemplo, o usuário comunica o desejo "Quero comprar o livro 'O Código Da Vinci'" e a infra-estrutura poderá incluir sugestões de compra de produtos relacionados, tais como, outros livros do mesmo autor, DVD – o filme, DVD multimídia com pinturas e documentos sobre Leonardo Da Vinci, reproduções de pinturas do pintor, ou mesmo pinturas originais do pintor, dependendo do perfil de compra e renda do usuário. Portanto, o usuário interage com a infra-estrutura que, baseada na descrição do usuário,

obtém de seus repositórios, os termos e outros itens relacionados e, se necessário, apresenta outras questões ao usuário para delimitar o escopo do serviço a ser realizado.

Para o exemplo da perda de carteira de identidade, nas interações iniciais com o usuário, algumas das regras de negócio poderiam ser expressas como:

- Se "usuário perdeu carteira de identidade" então deve ser apresentada a questão "Deseja solicitar a Segunda Via da identidade?".
- Se "usuário não quer solicitar a Segunda Via da Identidade" então deve ser apresentado o alerta "Você precisa registrar o BO urgentemente para que seu RG seja cancelado e evitar uso indevido".
- Se "usuário quer solicitar a Segunda Via da Identidade" então deve ser apresentado o procedimento para a solicitação.

Assim, a interação com o usuário, o diálogo, seria orientado por estas regras de negócio.

3.3.2 Metadados de Negócio – Vocabulários e Regras de Negócio

Segundo O'Neil [53], informações de contexto são fundamentais para um correto e efetivo uso dos dados de negócio. A linguagem natural é repleta de diferentes nuances nos significados das palavras, que podem ter diferentes significados dependendo do contexto em que são usadas. Os metadados adicionam informações de contexto aos dados de negócio definindo de maneira clara e explícita os significados e os relacionamentos entre esses dados. O próprio termo "termo" anteriormente citado também deve ser devidamente definido: refere-se a uma palavra ou frase que tem um significado preciso para o negócio e esse significado deve ser consistente tanto nos mecanismos de armazenagem (repositório de vocabulário) quanto nos valores associados às suas ocorrências no mundo real dos negócios.

Dito isto, todos os termos, regras de negócio e questões, anteriormente citados, devem ser estruturados em repositórios de vocabulário. Estes repositórios contêm os metadados com detalhes descritivos sobre os modelos de informação que representam. Neste caso, existem dois tipos principais de metadado: metadado de negócio que é capturado durante a análise de negócio, e metadado técnico que é capturado durante a fase de projeto e

implementação. Os dois tipos de metadado têm relação muito forte entre si e são disponibilizados para a comunidade de negócios da organização.

Todos os termos citados anteriormente, tais como “carteira de identidade”, “BO”, “Segunda Via da Identidade”, “solicitação de certidões negativas”, “solicitação de licenças”, “livro”, “autor”, “DVD”, “pintor”, etc. são termos que caracterizam-se como metadados de negócio e podem ser classificados por domínio de aplicação ou comunidade de negócio. Assim, os termos “carteira de identidade”, “BO” e “Segunda Via da Identidade” devem fazer parte do vocabulário da comunidade de identificação civil e os termos “livro”, “autor”, “DVD” e “pintor” devem fazer parte do vocabulário da comunidade de livrarias ou lojas afins. Estes vocabulários podem ser estruturados e gerenciados na forma de repositório conforme proposto neste trabalho. Da mesma forma, as regras de negócio relacionadas com a carteira de identidade citadas anteriormente são regras que se caracterizam como metadados de negócio e podem ser categorizados no repositório de regras da comunidade de identificação civil.

3.3.3 Metadados de Negócio – Ontologia

A realização de um serviço no nível dos negócios pode compreender a execução combinada de um conjunto de componentes de serviço. Para que esta execução combinada seja consistente é fundamental que os componentes de serviço sejam cuidadosamente descobertos, em tempo de modelagem ou de execução, e executados de forma colaborativa.

A polêmica em torno da descoberta dinâmica de serviços Web, em tempo de execução, já existe há alguns anos e, aparentemente, não terminará em breve. Nesta discussão, dentre as várias iniciativas as com maior aceitação parecem ser as levadas avante pela OASIS [54] com o UDDI [55] e DAML [56]com OWL-S [57]. Estas iniciativas, cada qual à sua maneira, procuram dar respostas ao desafio de descobrir os serviços, baseado em descrições sobre os mesmos, publicados em mecanismos de registro de serviços, mas nenhuma delas apresenta uma solução efetiva para o problema. Se a descoberta de serviços em tempo de execução ainda apresenta deficiências a invocação dos serviços, com todos os aspectos que

envolvem, principalmente, tratamento de erros, segurança e confiabilidade, é outro grande desafio a ser enfrentado [58, 59].

Ao invés de deixar os serviços totalmente "soltos" e desconectados um dos outros, propõe-se organizá-los por comunidades de negócio e formalizar as relações na forma de ontologias. Note-se que esta estruturação tem sentido somente para a comunidade em questão e é feita de forma abstrata, no sentido de que todos os serviços continuarão "soltos" e desconectados, do ponto de vista de outras comunidades de negócio. Assim, embora não totalmente dinâmica, em tempo de execução, porque esta organização ontológica é feita a priori, esta abordagem apresenta a vantagem de reduzir o tempo de descoberta porque reduz o escopo de pesquisa e, conseqüentemente, reduz o número de *hits* a serem considerados.

Conforme verbete em Wikipedia:Ontology [60]:

> "Uma ontologia é a tentativa para formular um esquema conceitual exaustivo e rigoroso dentro de um determinado domínio, uma estrutura de dados tipicamente hierárquica contendo todas as entidades relevantes, os relacionamentos e as regras dentro daquele domínio"

Em outras palavras, uma ontologia é uma descrição explícita e formal de conceitos e das relações entre estes conceitos em um universo de discurso. Ela provê um modo para organizar a formulação de questões e a conciliação semântica em sistemas computacionais. Ao capturar a estrutura e a semântica possibilita que uma máquina virtual de pesquisa baseada em ontologia possa controlar consultas simples através de palavras-chave e consultas complexas em dados estruturados.

Neste contexto, os serviços, que, individualmente, realizam parte dos processos de negócio de uma comunidade podem ser agrupados em conjuntos de serviços devidamente relacionados entre si. Os elementos e recursos usados para garantir esta categorização dos serviços e seus relacionamentos são também chamados de metadados de negócio e podem ser organizados numa ontologia de serviços. No Capítulo 4 descreve-se o Delimitador de Escopo de Serviço fazendo uso do Repositório de Ontologias de Serviços, cujas ontologias foram organizadas a priori, no instante de modelagem das regras de negócio.

3.3.4 Mecanismos de Regras de Negócio

Esta seção apresenta os principais conceitos e fundamentos relacionados com os mecanismos de regras de negócio, nos quais esta abordagem se baseia. Nesta proposta, a execução das regras e a execução dos componentes de serviço que compõem um serviço no nível dos negócios depende de fatos ou eventos associados aos negócios. Isto é feito com o uso de mecanismos de regras de negócio para gerar os fatos e avaliar as regras de negócio, que, baseados na existência ou não de fatos, executam ações que, por sua vez, invocam a execução dos componentes de serviço.

Um mecanismo de regras de negócio apóia-se em uma **base de conhecimento** e em um **procedimento de inferência** para deduzir novos fatos e atuar, a cada instante, de acordo com o estado desta base de conhecimento. A **base de conhecimento** representa explicitamente fatos (conhecimento estático do sistema) e regras (conhecimento dinâmico do sistema) expressos na forma de **sentenças**. As regras usam aqueles fatos como base para a realização de ações.

No mecanismo de regras são implementados técnicas e estratégias de raciocínio, de busca e resolução de conflitos. Conforme Barr [61], os mecanismos de regras geralmente adotam uma das seguintes estratégias de raciocínio:

- Encadeamento progressivo (*forward chaining)* – Também chamado de encadeamento dirigido por dados, parte de fatos conhecidos e tenta deduzir novos fatos, até chegar à solução. Ou seja, o lado esquerdo (LHS – Left Hand Side) da regra é examinado e, se for válida, o lado direito (RHS – Right Hand Side) desta é executado.
- Encadeamento regressivo (*backward chaining)*: Também chamado de encadeamento dirigido por objetivos. O sistema faz o caminho inverso, partindo da solução do problema (objetivo), tenta verificar se são verdadeiras todas as suas condições, que passam a ser então submetidas à verificação de suas respectivas condições. Isto ocorre sucessivamente até chegar-se a um conjunto de condições verdadeiras. Em outras palavras, os atributos dados como entrada são comparados

com os atributos do lado direito da regra e as regras que têm atributos iguais são selecionadas.

Para Chorafas [62] a combinação dessas duas estratégias, particularmente quando o espaço de busca é grande, mostra-se como uma boa solução.

O mecanismo de regras pode também utilizar princípios da lógica monotônica ou não monotônica, conforme possibilite a revisão de fatos no decorrer do processo de inferência. O raciocínio monotônico prevê que um fato, uma vez estabelecido na base de conhecimento, não poderá mais ser alterado. Já o raciocínio não monotônico permite este tipo de alteração.

Neste trabalho utiliza-se a combinação das duas estratégias de raciocínio porque o uso exclusivo do encadeamento progressivo dificulta o tratamento das regras que tenham um alinhamento muito forte com as necessidades (objetivos ou desejos) do usuário. No mundo dos negócios, uma vez executada uma ação, quase sempre é necessário inibir que esta mesma ação seja executada novamente, então o fato gerador da ação precisa deixar de existir para evitar inconsistências nos negócios. Assim, neste trabalho utilizam-se os princípios da lógica não monotônica.

Um mecanismo de regras pode estar gerenciando centenas ou milhares de regras. Num dado instante é provável que haja mais de uma regra que possa ser disparada pelos mesmos fatos. O conjunto de regras candidatas a serem disparadas é chamado de **conjunto de conflito** e o processo de ordenar as regras candidatas na ordem de disparo é chamado de **resolução de conflito**. Este processo de resolução de conflito produz uma lista ordenada de regras candidatas chamada de **Agenda.** Com relação à fase de resolução de conflitos, existem diversas possibilidades para se decidir qual das regras de um conjunto de conflito deve ser disparada. Exemplos dessas estratégias podem ser encontrados nos trabalhos de Russell e Norvig [63] e Rich e Knight [64]. Algumas possibilidades são:

- **recentidade**: disparar primeiro a regra ativada mais recentemente (*depth-first*);
- **Antigüidade**: disparar as regras na ordem na qual elas foram ativadas, isto é, primeiro a que está no conjunto de conflito há mais tempo (*breadth-first*);
- **especificidade**: disparar primeiro a regra mais específica, isto é, a de maior número de antecedentes;

- **prioridade**: disparar as regras de acordo com uma prioridade pré-estabelecida para as ações presentes no conseqüente;
- **não-duplicação**: não disparar a mesma regra com os mesmos argumentos duas vezes.

Em algumas situações a estratégia utilizada para a resolução de conflitos não tem muita influência no resultado, mas em outras situações a estratégia escolhida é importante. Cada uma destas estratégias isoladamente apresenta vantagens e desvantagens, porém a maioria dos mecanismos de regras permitem a combinação de algumas destas estratégias para obter uma melhor eficiência na resolução de conflitos.

No contexto deste trabalho, tendo como base os argumentos de Lindgren [65], considera-se que a combinação das estratégias de **recenticidade, prioridade e não duplicação** bastante adequada para garantir uma boa qualidade na resolução de conflitos. O uso do conceito de conjunto de regras *(rule set)* limitado por comunidades de negócio ajuda a reduzir drasticamente o **conjunto de conflito** neste trabalho. A inclusão de prioridade no disparo de algumas regras compensa a fraqueza da estratégia de recenticidade, a qual pode provocar o não disparo de regras incluídas ha mais tempo no conjunto de conflito. A estratégia de não-duplicação aliada ao princípio da lógica não monotônica garantem que, por exemplo, numa aplicação bancária, um saque não seja realizado duas ou mais vezes a partir dos mesmos fatos e dados.

3.3.5 O Mecanismo de Regras Jess

Na medida em que a execução de regras de negócio depende de um mecanismo de regras, apresenta-se a seguir uma pequena introdução sobre Jess, que é o mecanismo de regras usado na implementação nesta prova de conceito.

Embora uma regra em Jess [9] tenha semelhanças com um comando "se... então" de uma linguagem procedimental, ela não é usada de modo procedimental. Enquanto um comando "se... então" é executado em um momento específico e em uma ordem específica, uma regra em Jess é executada sempre que os padrões de sua parte "se" (o lado esquerdo do comando, também referido como LHS de *Left Hand Side* em Inglês) são satisfeitos. Para

isto é necessário que a máquina de regras do Jess esteja em execução. Isto faz com que as regras em Jess sejam menos determinísticas que um típico programa procedimental. Regras em Jess são definidas pela construção "defrule".

As regras em Jess têm duas partes, separadas pelo símbolo "=>" (que pode ser lido como "então") A primeira parte consiste nos padrões de LHS que são usados para “casar” com fatos da base de conhecimento e a segunda parte consiste na ação de RHS que pode conter chamadas de função.

Um padrão básico de uma regra em Jess pode incluir vários tipos de predicados, comparações e funções booleanas. Uma variável pode ser especificada para “casar” com qualquer valor naquela posição dentro de um padrão. Por exemplo, a regra:

```
(defrule exemplo-1
   (aa ?x ?y)
   =>
   (printout t "Saw 'aa " ?x " " ?y "'" crlf))
```

será ativada sempre que fatos, tais como (aa b c), (aa 1 2), (aa um um) forem definidos ou criados na base de conhecimento, ou seja, fatos com aa na cabeça da lista e mais dois valores. Em outro exemplo mais próximo da realidade, o padrão “(condutor_tem_habilitacao ?nomeCondutor ?numeroCNH)” tem como variáveis”?nomeCondutor” e “?numeroCNH”.

```
(defrule exemplo-2
   (condutor_tem_habilitacao ?nomeCondutor ?numeroCNH)
   =>
   (printout t "Condutor " ?nomeCondutor " tem habilitação " ?numeroCNH crlf))
```

Esta regra será ativada toda vez que qualquer fato com a cabeça “condutor_tem_habilitacao” e dois campos for incluída na memória de trabalho. Assim, fatos definidos ou criados na base de conhecimento, tais como “(condutor_tem_habilitacao João 123)”, “(condutor_tem_habilitacao 1 2)” ou “(condutor_tem_habilitacao um dois)” ativarão a ação de impressão acima. Porém, “(condutor_tem_habilitacao João)” e “(condutor_tem_habilitacao 123)”, não ativarão, porque apresentam somente um campo. Conforme pode ser visto nestes exemplos, as variáveis mapeadas nos padrões (LHS) de uma regra estarão disponíveis nas ações (RHS) da mesma regra.

O LHS de uma regra (a parte "se") consiste em padrões que devem ser “casados“ com fatos na base de conhecimento, e não chamadas de função. As ações de uma regra (a cláusula "então" ou RHS) é composto de chamadas de função.

Por exemplo, supondo que no contexto de locação de automóveis tenha-se a regra de negócio, em linguagem natural, a seguir:

> **É obrigatório uma** locação ser aprovada **somente se** locação está garantida **por um** cartão de crédito **e** condutor tem habilitação.

Assumindo-se que numa operação de locação façam sentido as variáveis para identificar a locação em si, o cartão de crédito, o condutor e a Carteira Nacional de Habilitação, esta regra poderia ser representada em Jess da seguinte forma:

```
(defrule rule_aprovaLocação
  (locação_garantida_por_cartão de crédito ?numeroLocação ?numeroCC)
  (condutor_tem_habilitacao ?nomeCondutor ?numeroCNH)
  =>
  aprovarLocação(numeroLocação)
)
```

3.4 Padrões e Tecnologias em Regras de Negócio

Os esforços na busca de uma **forma fácil, rápida e segura de refletir nas aplicações as mudanças** que ocorrem no dia a dia de qualquer organização, têm mobilizado a comunidade de pesquisa, organismos de padronização e empresas. Nas comunidades de pesquisa e nas empresas, nota-se um esforço na disponibilização de soluções integradoras baseadas nos serviços Web, nas máquinas de composição e orquestração de serviços e nas mecanismos de regras de negócio. Entre os organismos de padronização, embora partindo-se de iniciativas independentes percebe-se uma tentativa de convergência de esforços, notadamente entre a OMG [21], a W3C [22], a OASIS [54], e a BPMI [66], nas áreas de gerenciamento de processos de negócio, gerenciamento de serviços Web, gerenciamento de conhecimento e regras de negócio e em especificação de linguagens e ferramentas aderentes aos padrões e metamodelos propostos. As ramificações das pesquisas, com forte enfoque em desenvolvimento de software, abrangem principalmente as áreas de Sistemas Distribuídos, Engenharia de Software e algumas sub-áreas da Inteligência Artificial que tratam de Lógica, Mecanismos de Regras e Ontologia.

A Figura 3.1 mostra a evolução de alguns padrões e linguagens, derivados a partir das onipresentes linguagens XML [67] e UML [68], que podem ser usados para representar regras de negócio, ou que estão convergindo para representá-las. O acompanhamento desta evolução fornece uma visão geral sobre as principais características que podem ser úteis para representação semântica de regras de negócio e sobre aspectos em torno de metamodelos.

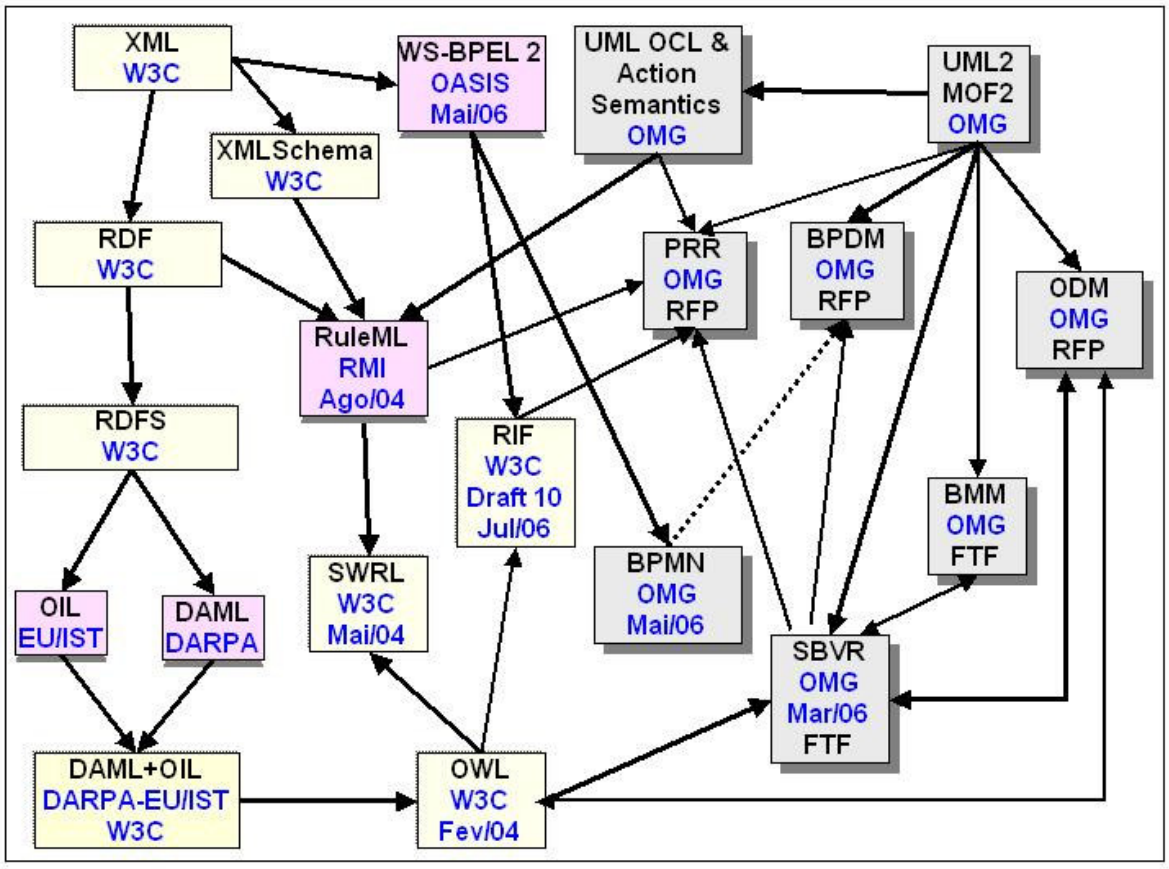

Figura 3.1 - Padrões e Linguagens em direção a Regras de Negócio.

As sub-seções a seguir apresentam visões gerais de alguns desses metamodelos e linguagens que se relacionam com as idéias desenvolvidas neste trabalho.

3.4.1 A Linguagem XML

XML é a abreviação para e**X**tensible **M**arkup **L**anguage [67] (Linguagem extensível de marcação) e é uma especificação desenvolvida pela W3C (World Wide Web Consortium),

para superar algumas limitações técnicas do HTML, que é o padrão das páginas da Web. A linguagem XML é definida como um formato universal de dados estruturados na Web.

XML tem algumas semelhanças com HTML, sendo a principal o fato de ambas utilizarem marcadores (*tags)*. Marcadores são palavras-chaves com função delimitadora para definir blocos de dados e que podem também conter parâmetros. Assim, o significado dos marcadores e fica a critério do programador. Portanto, e podem significar pedido, pessoa, nome, endereço, carro, enfim, o que o programador quiser que represente. Por essa característica, o XML é até considerado por muitos uma linguagem capaz de gerar outras linguagens, visto que quem define os comandos e suas funções é o programador. Esta praticidade possibilita que um usuário crie uma coleção própria de marcadores e aplique-os nas páginas e documentos que desejar.

Entre as funções principais do XML, destacam-se:

- descrever dados;
- apresentar dados em algum formato, como HTML;
- transportar dados;
- trocar dados de forma transparente entre plataformas diferentes.

É uma linguagem de muito sucesso em praticamente todas as áreas da Web. Através de um exemplo de pedido de compra, a Tabela 3.1 mostra algumas das facilidades de XML.

Tabela 3.1 - Representação XML de um Pedido de Compra.

```
...
<pedido data="20-11-2006">
    <endereçoDeRemessa país="BR">
        <nome>José Silva</nome>  <rua>Rua Nove, 1</rua>
        <cidade>Campinas</cidade>  <estado>SP</estado>  <cep>13080470</cep>
    </endereçoDeRemessa>
    <endereçoDeCobrança país="BR">
        <nome>Maria Silva</nome>  <rua>Rua Dez, 2</rua>
        <cidade>São Paulo</cidade>  <estado>SP</estado>  <cep>00123987</cep>
    </endereçoDeCobrança>
    <comentário>Urgente!</comentário>
    <itens>
        <item codigoProduto="872-AA">
            <nomeProduto> Lavadora de Roupas</nomeProduto>
            <quantidade>1</quantidade>
            <preço>500,00</preço>
        </item>
        <item codigoProduto="926-AA">
            <nomeProduto>Forno Microondas</nomeProduto>
            <quantidade>1</quantidade>
```

```
            <preço>300,00</preço>
        </item>
</itens>
</pedido>
...
```

3.4.2 A Linguagem OWL

OWL [69] é o acrônimo para *Web Ontology Language*, ou seja linguagem de ontologia para Web. Definida como uma recomendação da W3C para descrever semântica, foi projetada para garantir maior interpretabilidade computacional de conteúdos da Web do que a garantida por XML, RDF [70], e RDF Schema [71] provendo vocabulário adicional junto com uma semântica formal. Detalhes podem ser encontrados em *OWL Web Ontology Language Overview*.

Com OWL é possível representar declarações como "Tendo-se uma classe **Animal**, é possível dizer que **Macho** é uma **subclasse de Animal** e que **Fêmea** é uma **subclasse de Animal** e é uma **subclasse disjunta** de **Macho**", o que não é possível com RDFS. É possível também definir propriedades e sub-propriedades de objetos com domínio e intervalo para estas definições. A Tabela 3-2 mostra um exemplo de representação OWL para esses aspectos.

Tabela 3.2 – Aspectos representáveis em OWL que não são possíveis em RDFS.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<rdf:RDF
xmlns:owl ="http://www.w3.org/2002/07/owl#"
xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd ="http://www.w3.org/2001/XMLSchema#"
xmlns =""
>
…
    <owl:Class rdf:ID="Animal">
      <rdfs:label>Animal</rdfs:label>
    </owl:Class>

    <owl:Class rdf:ID="Male">
       <rdfs:label>Male</rdfs:label>
       <rdfs:subClassOf rdf:resource="#Animal"/>
    </owl:Class>

    <owl:Class rdf:ID="Female">
       <rdfs:subClassOf rdf:resource="#Animal"/>
       <owl:disjointWith rdf:resource="Male"/>
```

```
    </owl:Class>

    <owl:ObjectProperty rdf:ID="hasParent">
        <rdfs:domain rdf:resource="#Animal"/>
        <rdfs:range rdf:resource="#Animal"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:ID="hasFather">
        <rdfs:subPropertyOf rdf:resource="#hasParent"/>
        <rdfs:range rdf:resource="#Male"/>
    </owl:ObjectProperty>
...
</rdf:RDF>
```

3.4.3 A Linguagem RuleML

A meta da Rule Markup Initiative [41] é desenvolver RuleML [72] como uma linguagem canônica para regras na Web usando marcações XML e semântica formal com o intuito de obter implementações eficientes. A idéia é que RuleML possa cobrir um grande espectro de regras, desde regras de derivação até regras de transformação e regras de reação. Pretende-se que com RuleML seja possível especificar consultas e inferências a partir de ontologias na Web, especificar mapeamentos entre ontologias na Web, e especificar comportamentos dinâmicos na Web em workflows, serviços e agentes.

Ao invés de protótipos de pesquisa acadêmicos, RuleML procura focalizar em regras de interoperação entre padrões de indústria, tais como JSR 94 [44], SQL [73], OCL [74], Xlang [75], Xquery [76], RQL [77], OWL [69], OWL-S [78], e ISO Prolog [79]; como também em sistemas estabelecidos, como Blaze Advisor [7], ILOG Jrules [8], Jess [9], etc.

A Rule Markup Initiative desenvolve uma especificação modular para realizar transformações entre RuleML e outros padrões de máquinas de regra. Além disso, coordena o desenvolvimento de ferramentas para extrair, manter, e executar regras expressas em RuleML. Além da primeira versão de RuleML baseada unicamente em XML e da atual versão de RuleML combinando XML e RDF, há também uma abordagem para uma RuleML baseada unicamente em RDF. Outros esforços complementares consistem no desenvolvimento de máquinas de regras baseadas em Java, tais como Mandarax [80] RuleML, como também a XSB-RDF RuleML. Recentes esforços procuram definir uma sintaxe abstrata de RuleML como um Modelo MOF, chamado de MOF-RuleML.

Para exemplificar o uso de RuleML, no contexto de locação de automóveis, considerar que os estados de um carro incluem: 'disponível', 'reservado para aluguel', 'alugado', 'danificado', etc. A locadora pode usar estes nomes de estado para definir regras de negócio, tal como em "Um carro deve estar ´disponível´ ou ´reservado para aluguel´ antes da data de retirada". Os estados podem ser expressos também através de predicados unários, tal como em "carro está disponível." A Figura 3.2 mostra uma anotação OCL (*Object Constraint Language)* para o atributo (estado) isAvailable da classe RentalCar, extraído de [64].

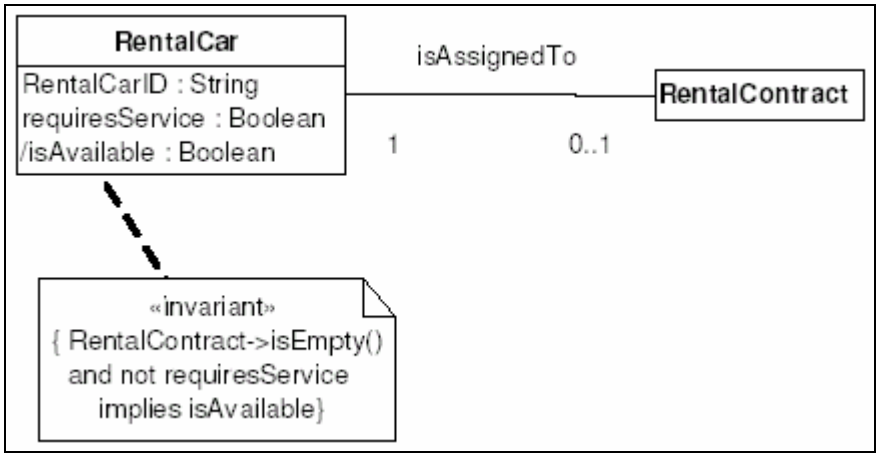

Figura 3.2 - Invariante OCL para o atributo derivado isAvailable da classe RentalCar.

O trecho de código RuleML na Tabela 3-3 representa esta anotação OCL. Note que _head indica a conclusão (equivalente ao lado esquerdo numa regra de produção) e _body indica a condição de uma regra de derivação em RuleML. Um tipo de negação em RuleML é expresso pela *tag* <naf> que significa "*negation-as-failure*".

Tabela 3.3 - Codificação em RuleML, de Invariante OCL.

```
<imp>
<_head>
   <atom>
      <_opr>isAvailable</_opr>
      <var>Car</var>
   </atom>
</_head>
<_body>
   <atom>
      <_opr>RentalCar</_opr>
      <var>Car</var>
   </atom>
   <naf>
```

```
            <atom>
               <_opr>requiresService</_opr>
               <var>Car</var>
            </atom>
         </naf>
         <naf>
            <atom>
               <_opr>isAssignedToRentalContract</_opr>
               <var>Car</var>
            </atom>
         </naf>
      </_body>
</imp>
```

Fraquezas de RuleML

- Não é possível incluir variáveis nas premissas das regras, o que obriga o programador a fazer "malabarismos" para contornar esta situação.
- Expressar regras é difícil e excessivamente verboso [81].
- Não é possível expressar regras de reação com o formato evento-condição-ação-efeito
- O uso inadequado da *tag* <imp> (de implicação) para regras de derivação.

3.4.4 A linguagem SWRL

SWRL (*Semantic Web Rule Language*) [82] propõe combinar RuleML e OWL para acrescentar semântica a regras e ontologias. A proposta que combina as sub-linguagens de OWL (OWL DL e OWL Lite) com a sub-linguagens de RuleML (Unary/Binary Datalog RuleML) estende o conjunto de axiomas de OWL com regras na forma de cláusulas de Horn.

As regras propostas são da forma de uma implicação entre um antecedente (corpo) e conseqüente (cabeça). Átomos múltiplos são tratados como conjunção. Átomos nestas regras podem ser da forma *C(x), P(x,y), sameAs(x,y) ou differentFrom(x,y)*, onde *C* é uma descrição OWL, *P* é uma propriedade OWL, e *x,y* são variáveis quaisquer, indivíduos OWL ou valores de dados OWL.

Com SWRL é possível representar expressões simples que não são possíveis em OWL, tais como "dizer que o irmão de meu pai é meu tio". Assim, a expressão em Inglês "*Your*

parent´s brothers are your uncles" pode ser expressa em SWRL, conforme mostra a Tabela 3-4.

Tabela 3.4 - Código SWRL para "*Your parent´s brothers are your uncles*" [82]

```
...
<swrl:Variable rdf:ID="x1"/>
<swrl:Variable rdf:ID="x2"/>
<swrl:Variable rdf:ID="x3"/>
<ruleml:Imp>
 <ruleml:body rdf:parseType="Collection">
  <swrl:individualPropertyAtom>
   <swrl:propertyPredicate rdf:resource="&eg;hasParent"/>
   <swrl:argument1 rdf:resource="#x1" />
   <swrl:argument2 rdf:resource="#x2" />
  </swrl:individualPropertyAtom>
  <swrl:individualPropertyAtom>
   <swrl:propertyPredicate rdf:resource="&eg;hasBrother"/>
   <swrl:argument1 rdf:resource="#x2" />
   <swrl:argument2 rdf:resource="#x3" />
  </swrl:individualPropertyAtom>
 </ruleml:body>
 <ruleml:head rdf:parseType="Collection">
  <swrl:individualPropertyAtom>
   <swrl:propertyPredicate rdf:resource="&eg;hasUncle"/>
   <swrl:argument1 rdf:resource="#x1" />
   <swrl:argument2 rdf:resource="#x3" />
  </swrl:individualPropertyAtom>
 </ruleml:head>
</ruleml:Imp>
…
```

3.4.5 O Metamodelo PRR

O metamodelo PRR (Production Rule Representation) [83] está sendo desenvolvido num esforço conjunto de três iniciativas de padronização envolvidas na definição de representação de regras de produção independente de domínio, quais sejam (i) a OMG, representada pelo grupo *Business Modeling Integration* e desenvolvedores da resposta para o RFP [84]; (ii) a RuleML, uma família de padrões relacionados com regras no contexto da Web Semântica e incluindo uma RuleML baseada neste metamodelo PRR; e (iii) a W3C, que criou um grupo de trabalho para definir o RIF (Rule Interchange Format) [85], um formato para intercâmbio de regras.

O metamodelo PRR, que está ainda na fase de análise de propostas submetidas, tem como objetivo a representação de regras de produção em modelos de UML. Para isto, o

RFP [84] solicitou propostas para representação de regras de produção para definir (i) um metamodelo aderente a MOF2 [86] com semântica precisa para representar e explicitar regras de produção como elementos de modelos UML; (ii) um XMI (xsd - *XML Schema Description*) para regras de produção baseadas no metamodelo proposto, para facilitar a troca de regras de produção entre ferramentas de modelagem e máquinas de inferência; e (iii) um exemplo de sintaxe (não-normativa) aderente ao metamodelo proposto para expressar regras de produção em modelos UML.

3.4.6 A Especificação JSR 94

A Comunidade de Java também identificou a importância das máquinas de regra e produziu a especificação chamada de JSR 94 [44]. O JSR 94 especifica a terminologia básica das máquinas de regra, uma operação típica de uma máquina de regra e quais são as funcionalidades básicas de uma máquina de regra. Conforme pesquisas de Daniel Selman Johan Majoor [87], os mecanismos de regras mais populares e aderentes a esta especificação são Drools [88], CleverPath Aion da Computer Associates [89], Haley [90], Blaze Advisor da Fair Isaac [7], ILOG Jrules [8], Jess da Sandia Labs [9] e PEGARules da Pegasystems [91].

3.5 O Metamodelo SBVR - *Semantics of Business Vocabulary and Business Rules*

Devido à falta de consenso para definir regras na terminologia de negócios a OMG publicou um Pedido de Propostas (BSBR RFP – *Business Semantics of Business Rules* Request For Proposal) [92]. Este RFP solicitou propostas para o metamodelo SBVR (*Semantics of Business Vocabulary and Business Rules*) [19] que está em fase final de adoção pela OMG. O objetivo do metamodelo SBVR é permitir que pessoas de negócio definam as políticas e as regras que conduzem as organizações na própria linguagem destas pessoas de negócios, em termos dos artefatos com os quais elas realizam os negócios. Além

disso, o outro objetivo é capturar essas regras de um modo claro, não ambíguo, e rapidamente transformável em outras representações, como as representações para as pessoas de negócios, para os engenheiros de software, e para os sistemas automatizados de execução de regras de negócio. Este RFP pediu propostas para (i) um metamodelo MOF (*Metadata Object Facility*) [93] para a especificação de regras de negócio para pessoas de negócio; (ii) um metamodelo para a captura de vocabulários e definições das condições usada em regras de negócio; e (iii) uma representação XML de regras de negócio e vocabulários baseados em XMI [94] para permitir a interoperabilidade de regras e vocabulários entre ferramentas de software que manipulam regras de negócio. Deve ser ressaltado que o metamodelo SBVR resultante não é orientado para as pessoas de negócio e sim para engenheiros de software que constroem ferramentas para as pessoas de negócio, pois ele se situa no nível CIM [95] da arquitetura MDA [96], conforme sugere a OMG (Figura 3-13).

3.5.1 Beneficiários de SBVR

O SBVR provê diferentes perspectivas considerando os diferentes grupos de pessoas que poderão beneficiar-se do metamodelo. A seguir algumas destas perspectivas.

A visão do negócio na **perspectiva dos analistas de negócio** é a visão do negócio da organização ou de uma parte desta visão, se considerarmos que uma comunidade dentro da organização tem uma visão parcial do negócio da organização. Desta forma, os analistas de negócio estão interessados em sistemas construídos com base no vocabulário compartilhado pela comunidade, na qual estão acostumados a realizar seus negócios. Eles precisam negociar com gerentes a inclusão de novos conceitos, termos e regras de negócio no vocabulário e precisam especificar precisamente as políticas e regras de negócio. Porém, para isto, não precisam de qualquer conhecimento detalhado da formulação lógica de SBVR. Assume-se que o tratamento desta formulação lógica de SBVR será realizado por facilidades incorporadas nas ferramentas de suporte a vocabulários e regras de negócio, tais como, disponibilização de modelos (*templates)*, configuração de restrições, e verificação de consistência.

Da **perspectiva dos desenvolvedores de ferramentas,** o SBVR necessitará de dois tipos de ferramentas: (i) para intercâmbio de vocabulários e regras de negócio entre diferentes plataformas; e (ii) para desenvolver e manter vocabulários e regras de negócio para uma comunidade. Para a OMG, o intercâmbio de padrões é de grande importância e, por isso, o principal requisito em relação ao desenvolvimento de ferramentas aderentes a SBVR foi a garantia de compatibilidade e aderência a MOF e XMI. Em relação às ferramentas para desenvolvimento e manutenção de vocabulários e regras de negócio o principal requisito refere-se à necessidade de suportar o compartilhamento, entre múltiplas comunidades, de conjuntos de termos e expressões de negócio com significado precisamente definidos e a formulação lógica a partir destes termos e expressões. Além disso, outro aspecto importante que as ferramentas devem incorporar é um processo de desenvolvimento de vocabulários e regras de negócio de modo a assegurar a integração e o compartilhamento com outras comunidades.

Da **perspectiva dos especialistas em lógica, semântica e lingüística**, o SBVR provê capacidades lógicas, matemáticas, e lingüísticas que tornam possível transformar vocabulários e regras de negócio a partir de modelos de negócio para modelos PIM e PSM [96], estruturar uma variedade de sentenças em linguagem natural em construções SBVR e verbalizar estruturas SBVR em termos de sentenças em linguagem natural. Assim, é possível projetar algoritmos que assegurem a integridade de vocabulários e regras de negócio em documentos de intercâmbio e na tradução entre documentos de intercâmbio e modelos internos das ferramentas. Com isto, assegura-se também a formalização matemática e lógica dos modelos internos usados nas ferramentas de desenvolvimento de vocabulário e regras de negócio.

A abordagem proposta neste trabalho processa regras de negócio expressas em conformidade ao metamodelo SBVR.

3.5.2 Fundamentos de SBVR

A fundamentação de regras de negócio em SBVR tem suas origens na idéia defendida no “Manifesto de Regras de Negócio” [97], onde "Regras baseiam-se em fatos, e fatos baseiam-se em conceitos que são expressos por termos. Termos expressam conceitos de negócio; fatos fazem asserções sobre estes conceitos; regras restringem e dão suporte a estes fatos." Essa idéia básica, que surgiu em 1995 no âmbito do Business Rules Group [98], tem sido chamada de "mantra" de regras de negócio. Por conveniência, esse “mantra” é, freqüentemente, abreviado para "Regras são baseadas em fatos, e fatos são baseados em termos." A Figura 3-3, extraída de SBVR [19] procura mostrar como a especificação suporta a idéia desse "mantra."

O SBVR pode ser visto a partir de cinco grandes aspectos, conforme mostra o diagrama da Figura 3-4.

- Comunidade *(Community)* - A base para o vocabulário de negócio é a comunidade. No nível dos negócios, comunidades de importância primária são as próprias empresas para as quais as regras de negócio estão sendo estabelecidas e expressas. Porém, outras comunidades, tais como, o setor na qual a empresa opera, empresas parceiras, organizações de padronização, e autoridades reguladoras, também precisam ser consideradas.
- Conjunto de Significados Compartilhados *(Body of Shared Meanings)* - Uma comunidade tem um conjunto de elementos com significados compartilhados, compreendendo os conceitos, os fatos e as regras de negócio. Para que os significados compartilhados possam ser trocados, discutidos e validados, eles devem ser expressos. Porém, o que é compartilhado é o significado e não a forma de expressão. Assim, SBVR separa o significado do negócio de qualquer forma particular de expressão.

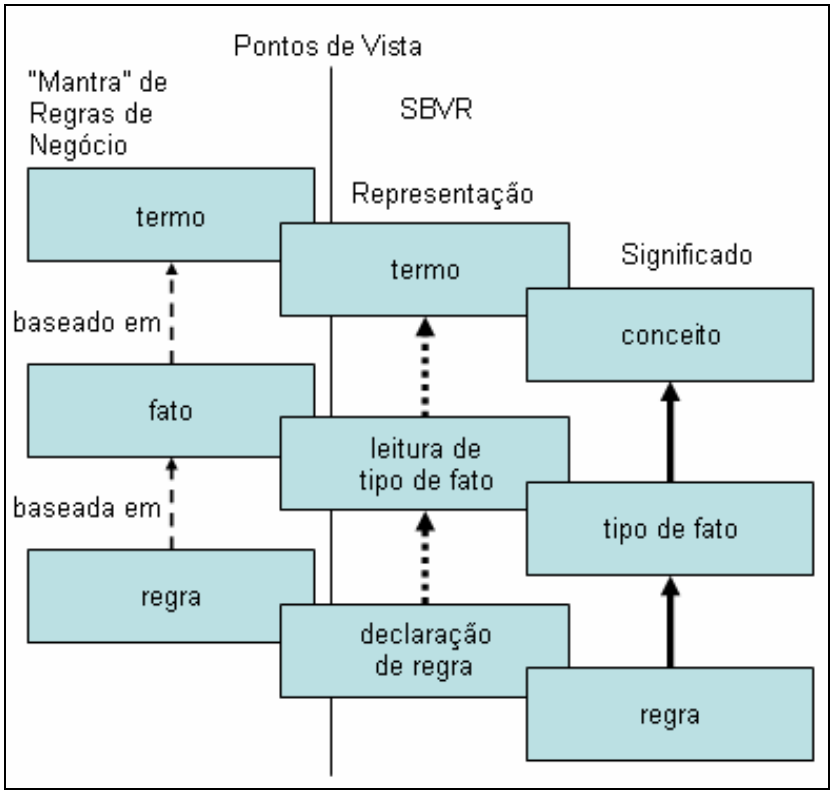

Figura 3.3 - SBVR suportando o "Mantra" de Regras de Negócio.

- Formulação Lógica *(Logical Formulation)* – A formulação lógica provê uma sintaxe formal, abstrata e independente de linguagem para capturar a semântica de um conjunto de elementos de significados compartilhados. Apóia-se em múltiplas formas de representação, tais como, substantivos e expressões verbais, com associações na voz ativa e na voz passiva. A formulação lógica suporta duas características essenciais de SBVR, quais sejam, (i) o mapeamento de um conjunto de elementos de significados compartilhados para vocabulários usados por comunidades; e (ii) o mapeamento para XMI que padroniza o intercâmbio de conceitos, fatos e regras negócio entre ferramentas que suportam o metamodelo SBVR.

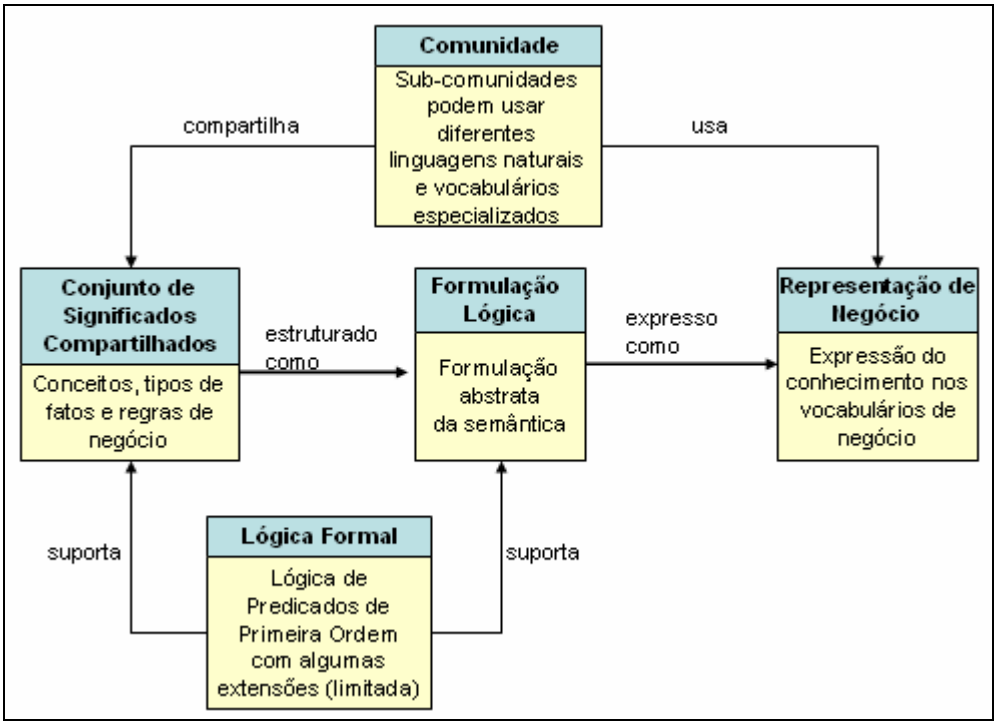

Figura 3.4 - SBVR em cinco aspectos.

- Representação de Negócio *(Business Representation)* - Os conceitos e as regras de negócio em um conjunto de elementos de significados compartilhados precisam ser representados em vocabulários aceitáveis e utilizáveis por comunidades que compartilham aqueles significados. Estes vocabulários podem estar em linguagens naturais diferentes, em linguagens artificiais como o UML, ou em subconjuntos especializados de linguagens naturais, usadas por exemplo, por engenheiros ou advogados. O SBVR suporta a representação de elementos de negócio associando símbolos para conceitos, por exemplo, para termos como "cliente", "carro", "usuário" e símbolos para fatos (freqüentemente expressões verbais) como "alugar", "fica situado a", "estar cancelado" ou “deve registrar”.
- Lógica Formal *(Formal Logic)* - A fundamentação teórica de SBVR baseia-se na lógica formal, suportando a formulação lógica e as estruturas dos elementos de significados compartilhados. A base é a lógica de predicados de primeira-ordem, com algumas extensões limitadas em lógica modal, para expressar obrigação, proibição, e necessidades.

3.5.3 Principais características de SBVR

Há vários metamodelos para representação de regras. A natureza da maioria desses metamodelos difere da natureza de SBVR. Apresenta-se a seguir uma visão geral das principais características de SBVR, conforme consta em SBVR Annex M [99].

- Nenhum padrão ainda é usado amplamente em termos comerciais.
- Com respeito às regras, estes metamodelos focalizam em representação, enquanto SBVR focaliza em significados discretos e únicos, independente de forma ou representação.
- SBVR inclui uma formalização teórica sobre modelos e formulações semânticas de vocabulários e regras de negócio.
- Somente SBVR provê cláusulas de necessidade e obrigação, que são críticas para a representação formal de regras de negócio.
- SBVR dá ênfase especial às cláusulas de obrigação. No mundo real de atividades de negócio as pessoas podem violar tais regras de negócio, um fato crucial que outros metamodelos não conseguem tratar adequadamente.

É possível criar transformações de SBVR para qualquer outro metamodelo ou vice-versa. O desenvolvimento de transformações deve considerar os pontos seguintes na transformação de SBVR para o outros metamodelos:

- Decidir como tratar cláusulas de necessidade e de obrigação nas regras. Uma opção é traduzi-las para predicados.
- Algumas das representações não aderentes a SBVR não têm um operador equivalente para os operadores SBVR '*whether or not*' e '*equivalence'*.
- Algumas das representações não aderentes a SBVR não têm os operadores equivalentes para quantificadores como '*each'*, '*some'*, '*at least one*,' etc. Neste caso é possível criar predicados especiais ou funções para lidar com esta semântica.

3.5.4 O vocabulário de negócio

Uma característica importante das regras de negócio é que elas podem e devem ser apresentadas às pessoas de negócio na sua própria terminologia. Ou seja, as regras de negócio devem ser inteligíveis para as "pessoas de negócio", que têm responsabilidade pelas atividades de negócio para as quais as regras se aplicam, e para as pessoas que entendem como as regras se relacionam aos objetivos do negócio. A sintaxe das regras pode ser textual ou gráfica, mas o vocabulário deve ser um vocabulário de negócio, significativo às pessoas que fazem e cumprem as regras de negócio, conforme especificado em BSBR-RFP [92].

Um "Vocabulário de negócio" é um vocabulário cujas definições consistem no entendimento compartilhado entre uma comunidade de pessoas de negócio sobre os artefatos com os quais eles tratam para realizar o negócio, e cujas palavras e frases, precisamente entendidas pela comunidade, referem-se unicamente a uma das definições dentro de um contexto do negócio.

Em SBVR usa-se designações e formas de expressões exatamente como eles estão definidos em um vocabulário. As formas no plural não são usadas. Assim, uma declaração formal diria "cada conceito" ao invés de "todos os conceitos." Se a forma ativa e a forma passiva são usadas então ambas precisam ser definidas em um vocabulário.

3.5.5 O Pacote "Essential SBVR"

O *'Essential SBVR package'* é um pacote UML/MOF que contém classes usadas por outros pacotes que contêm regras de mapeamento SBVR-MOF/XMI. Seus conteúdos são definidos usando declarações formais do *Essential SBVR Vocabulary* mostrado na Figura 3.5.

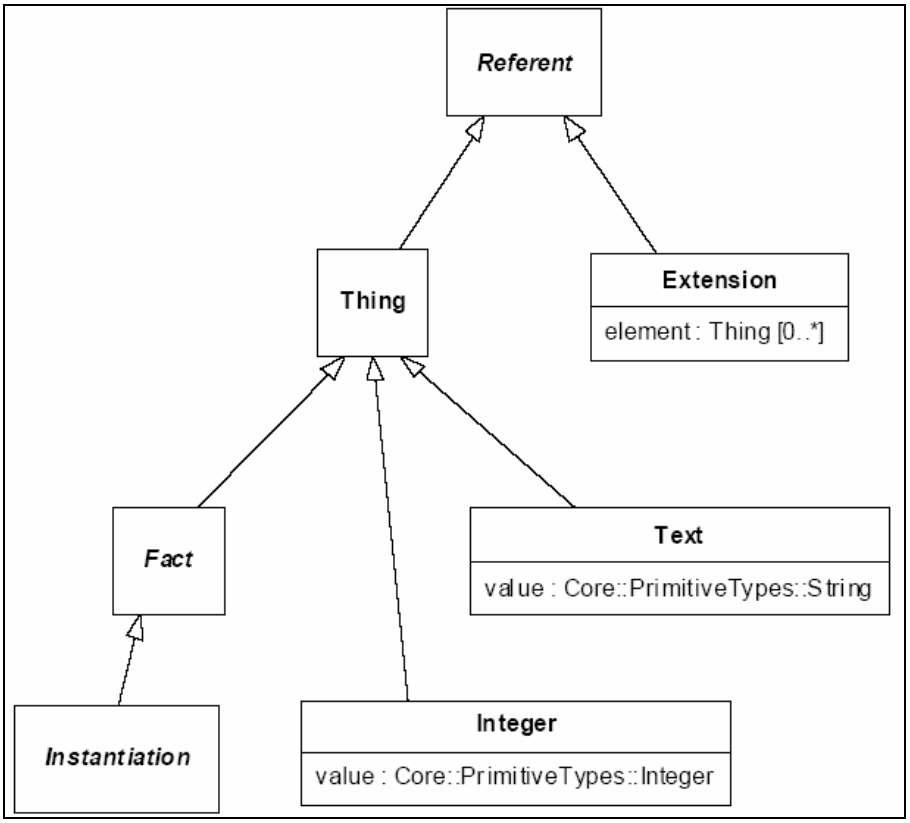

Figura 3.5 - O pacote Essential SBVR.

A classe ***Referent*** é usada para representar a quê um fato pode referenciar. Tem duas subclasses: (i) a classe ***Thing*** é usada para representar coisas individuais; e (ii) a classe ***Extension*** é usada para representar um conjunto de coisas, o conjunto inteiro para o qual um fato é verdade. Cada instância da classe ***Extension*** tem zero ou mais coisas que são instâncias da classe de ***Thing***.

A classe ***Thing*** é usada para representar coisas que são sujeitos ou objetos de fatos. O tipo ou tipos de uma coisa são conhecidas a partir de fatos sobre esta coisa. A classe ***Thing*** tem subclasses para representar diferentes tipos de coisas: a subclasses ***Fact***, ***Text*** e ***Integer***.

Uma subclasse ***Fact*** é criada de acordo com as regras de mapeamento SBVR-MOF/XMI para cada sentença num dado vocabulário. Cada instância desta subclasse representa um fato do tipo do fato associado à sentença.

Uma subclasse ***Instantiation*** é criada de acordo com as regras de mapeamento SBVR-MOF/XMI para cada termo em um vocabulário. Cada instância de uma destas subclasses representa um fato em que algo é classificado como sendo do tipo indicado pelo termo.

A classe ***Text*** e a classe ***Integer*** provêem meios convenientes para usar textos e inteiros na representação de fatos.

O metamodelo SBVR para especificação de regras de negócio é orientada a fatos. Esta abordagem permite que os próprios fatos possam ser sujeitos de outros fatos. Um fato é ele próprio um tipo de fato (*fact type*) e portanto, pode ser sujeito de outros fatos.

As definições de classes na Tabela 3-5, tais como ***Instantiation*** e ***Fact***, foram extraídas de SBVR e ajudam a ilustrar a estrutura do metamodelo para criar modelos específicos de uma comunidade.

Tabela 3.5 - Definição de parte do Pacote Essential SBVR.

Referent Class
Definition: the class that *is owned by* the **Essential SBVR Package** and that *has* the name 'Referent'
Necessity: The **Referent Class** *is abstract*.

Thing Class
Definition: the class that *is owned by* the **Essential SBVR Package** and that *has* the name 'Thing'
Necessity: The **Thing Class** *is* not *abstract*.
The **Referent Class** *is* a superclass *of* the **Thing Class**.

Fact Class
Definition: the class that *is owned by* the **Essential SBVR Package** and that *has* the name 'Fact'
Necessity: The **Fact Class** *is abstract*.
The **Thing Class** *is* a superclass *of* the **Fact Class**.
'fact' *is* the XMI name *of* the **Fact Class**.

Instantiation Class
Definition: the class that *is owned by* the **Essential SBVR Package** and that *has* the name 'Instantiation'
Necessity: The **Instantiation Class** *is abstract*.
The **Fact Class** *is* a superclass *of* the **Instantiation Class**.
'instantiation' *is* the XMI name *of* the **Instantiation Class**.

Integer Class
Definition: the class that *is owned by* the **Essential SBVR Package** and that *has* the name 'Integer'
Necessity: The **Thing Class** *is* a superclass *of* the **Integer Class**.
The **Integer Class** *is* not *abstract*.

3.5.6 Fatos e Conceitos em Regras de Negócio

A Figura 3.6 fornece uma idéia do metamodelo SBVR no que se refere ao tratamento de fatos e conceitos para representar regras de negócio. Como pode ser visto, um tipo de fato (*fact type*, à direita) em que se baseia um fato (*fact*, na parte inferior à direita) é fortemente baseado nas características de um conceito (*concept*, à esquerda) que por sua vez é expresso por termos que representam os objetos (*thing*, na parte inferior).

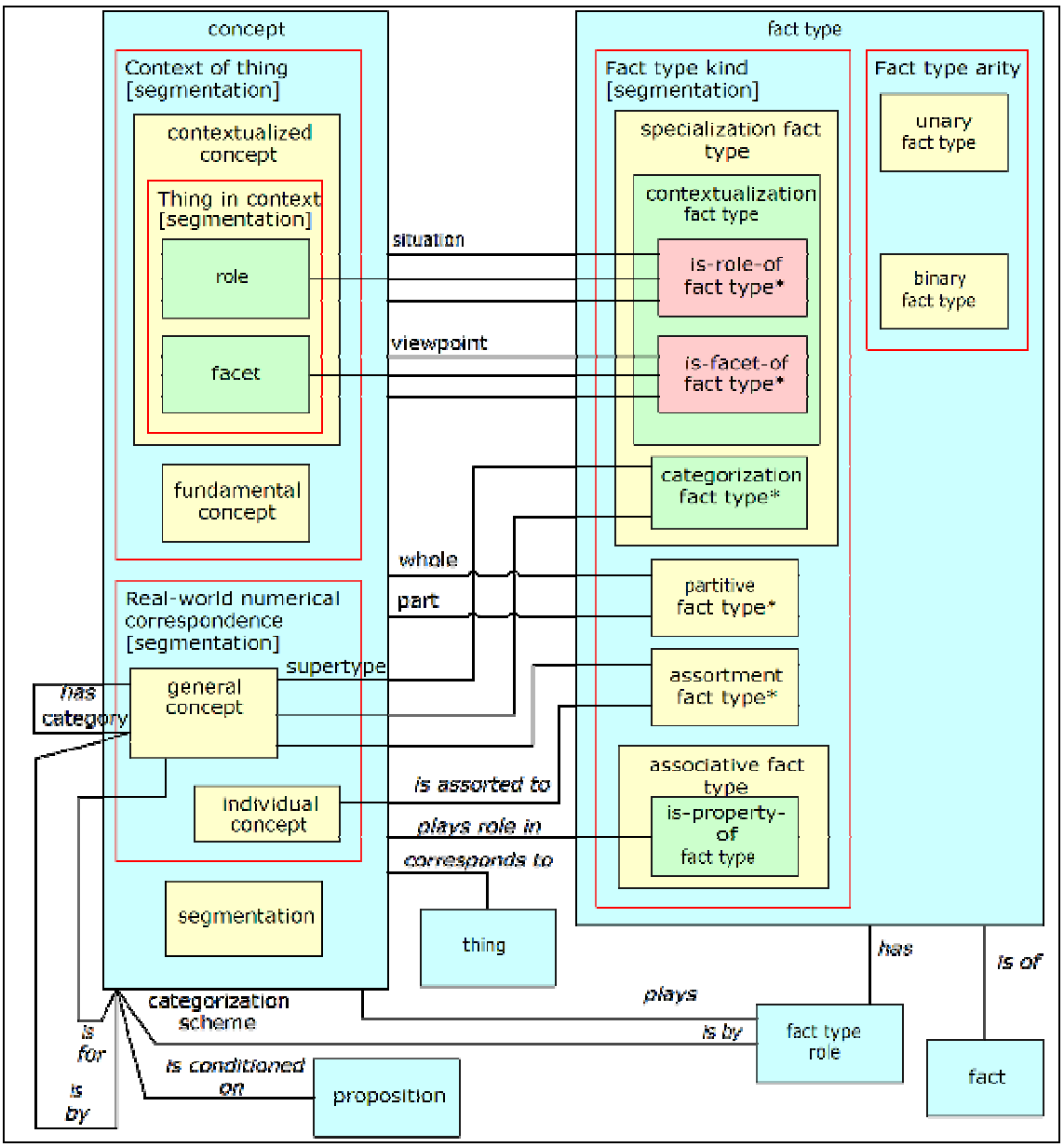

Figura 3.6 - Tratamento de fatos e conceitos em SBVR.

A seguir está a descrição extraída de SBVR para o elemento **binary fact type** contido no diagrama da figura anterior.

> **binary fact type**
> Definition: fact type that *has* exactly 2 roles
> Example: The fact type 'shipment has drop-off location' whose instances are actualities of shipments having drop-off locations.

O fato representado pela expressão "condutor devolveu o carro de aluguel" é do tipo binário (*binary fact type*), onde "condutor" e "carro de aluguel" são "*fact type role*"s.

3.5.7 Palavras-Chave em Sentenças SBVR

As principais palavras-chave SBVR para formulações lógicas em Inglês estruturado são mostradas a seguir. As letras 'n ' e 'm ' representam uso de um número inteiro e as letras 'p ' e 'q ' representam expressões de proposições.

Quantificação	
each	*universal quantification*
some	*existential quantification*
at least one	*existential quantification*
at least n	*at-least-n quantification*
at most one	*at-most-one quantification*
at most n	*at-most-n quantification*
exactly one	*exactly-one quantification*
exactly n	*exactly-n quantification*
at least n and at most m	*numeric range quantification*
more than one	*at-least-n quantification with n = 2*

Operações Lógicas	
it is not the case that p	*logical negation*
p and q	*conjunction*
p or q	*disjunction*
p or q but not both	*exclusive disjunction*
if p then q	*implication*
q if p	*implication*
p if and only if q	*equivalence*
not both p and q	*nand formulation*
neither p nor q	*nor formulation*
p whether or not q	*whether-or-not formulation*

Operações Modais	
it is obligatory that p	*obligation claim*
it is prohibited that p	*obligation claim embedding a logical negation*

it is necessary that p	*necessity claim*
it is impossible that p	*necessity claim embedding a logical negation*
it is possible that p	*possibility claim*
it is permitted that p	*permissibility claim*

As palavras-chave seguintes são usadas no meio de expressões, com verbos normalmente no infinitivo, para formar expressões verbais equivalentes a algumas das operações modais listadas na Tabela anterior.

Operações Modais	
… must …	*obligation claim*
… must not …	*obligation claim embedding a logical negation*
… always …	*necessity claim*
… never …	*necessity claim embedding a logical negation*
… may …	*permissibility claim*

A palavra-chave "only if" é usada em combinação com algumas das palavras-chave para operações modais para inverter a modalidade.

… may … only if p	*obligation claim over an implication*
it is permitted that q only if p	*obligation claim over an implication*
it is possible that q only if p	*necessity claim over an implication*

3.5.8 Representação de Fatos e Regras

Esta seção apresenta alguns exemplos de como fatos e regras de negócio aderentes ao metamodelo SBVR poderiam ser representados. Nestas representações, os termos estão sublinhados e as **palavras-chave** estão em **negrito**, para diferenciá-los dos verbos e expressões verbais que compõem os fatos, que estão em texto normal. A partir de um pequeno subconjunto do vocabulário do domínio de locação de automóveis extraído da proposta de SBVR pode-se definir os seguintes termos e fatos: branch has name, branch, name, branch owns car, car, car has vehicle identification number, vehicle identification number. O modelo UML/MOF correspondente para este vocabulário SBVR pode ser representado pelo diagrama da Figura 3-7.

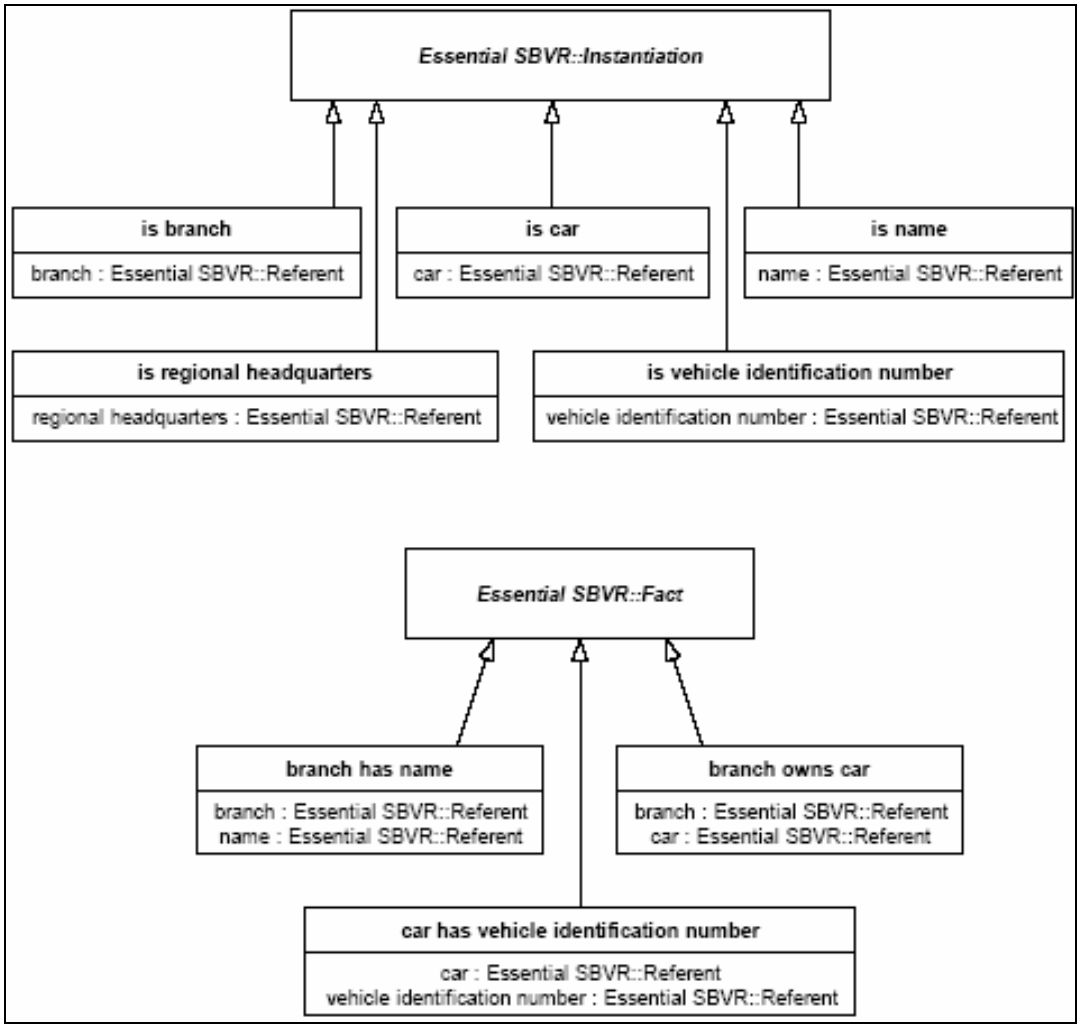

Figura 3.7 - Modelo UML/MOF para parte do vocabulário de EU-Rent.

As subclasses da classe *Instantiation* mostradas no diagrama acima são usadas para representar o fato de que um objeto é classificado como sendo do tipo correspondente. Assim, uma instância da classe carro não é usada para representar um carro e sim para representar o fato de que um objeto é classificado como sendo um carro. As subclasses da classe *Fact* são usadas para representar fatos baseados em sentenças do vocabulário. A partir deste modelo pode-se expressar fatos relacionados usando-se as seguintes sentenças.

A given car has **the** vehicle identification number 'ABC123'.

A given branch has **the** name 'London Downtown'.

The car is owned by **the** branch.

The branch is **a** regional headquarters.

Estes fatos podem ser representados em termos do modelo UML/MOF através do diagrama de instâncias da Figura 3-8.

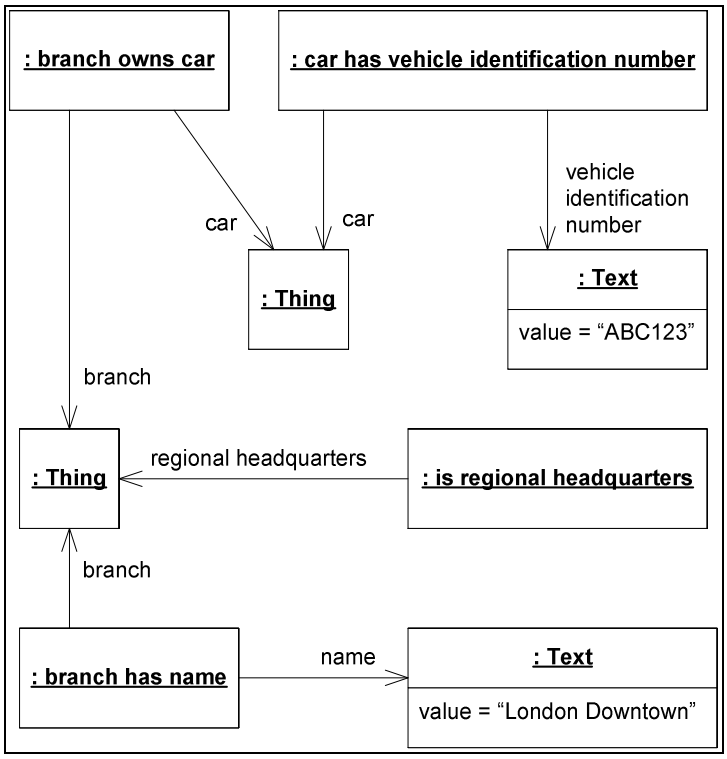

Figura 3.8 - Diagrama de Instância de fatos para parte do vocabulário EU-Rent.

Este exemplo é simples e representa alguns fatos individuais. Às vezes é desejável representar uma extensão inteira de um conceito. O exemplo da Figura 3-9 mostra que o conjunto de todas as sedes regionais são as filiais de nome “London Downtown”, “Tokyo Main Branch” e “New York Branch”.

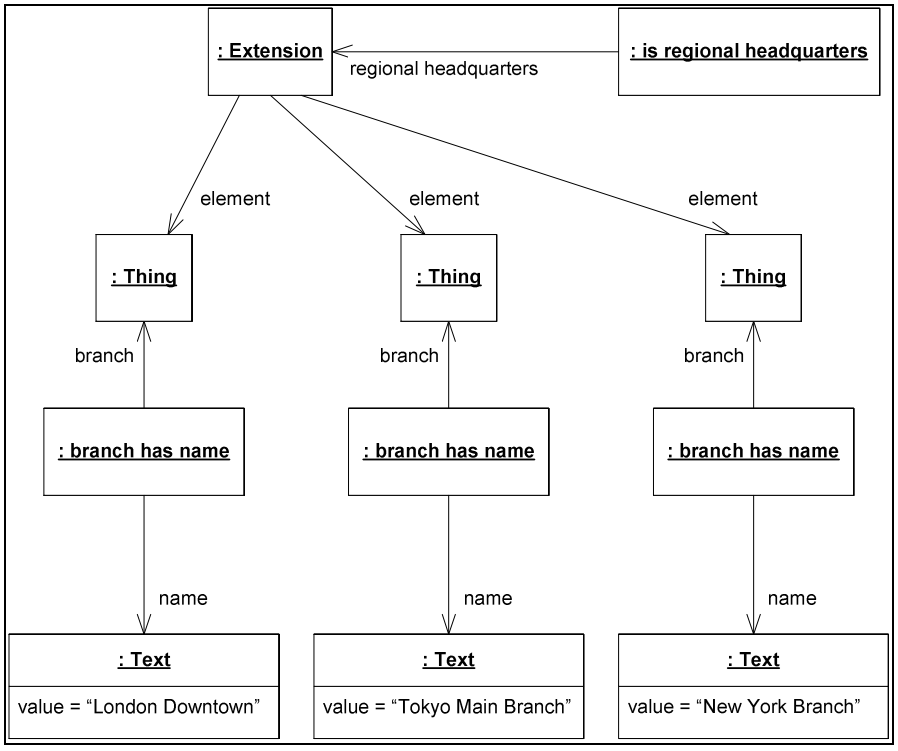

Figura 3.9 - Diagrama de Instância de fatos para parte do vocabulário EU-Rent.

A classe *Extension* também pode ser usada para representar tudo em relação a qualquer outra coisa. Por exemplo, a Figura 3-10 representa que todos os carros de propriedade da filial "London Downtown” são os com placa (*vehicle identification number*) ABC123 e XZY987.

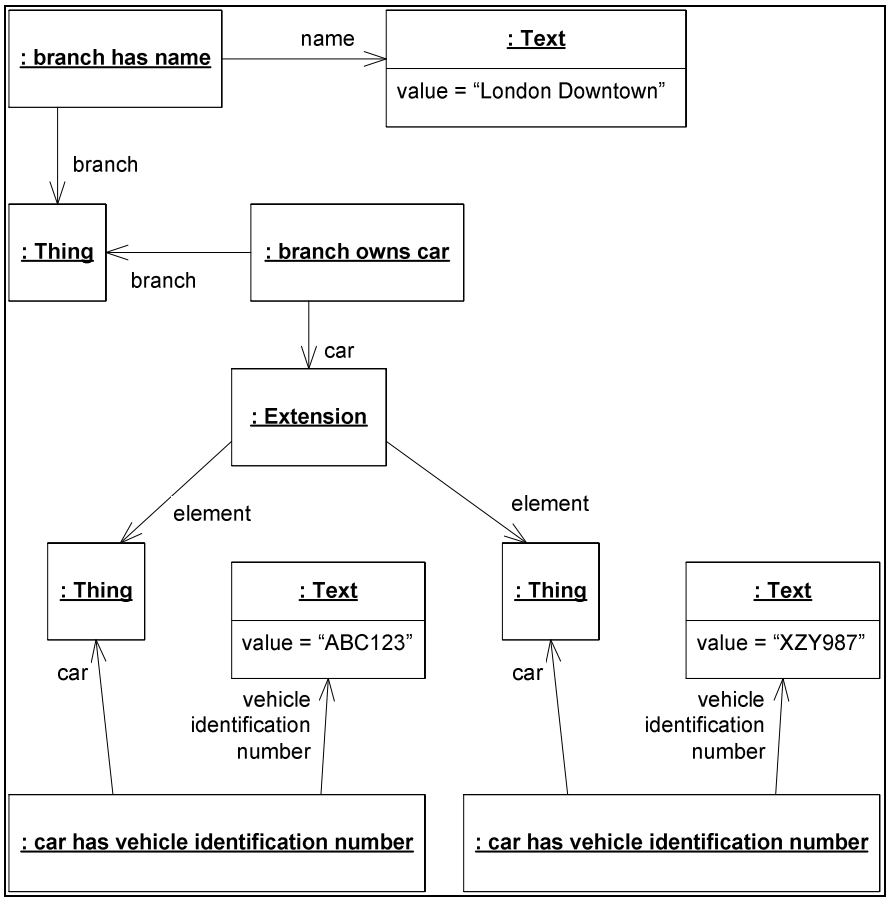

Figura 3.10 - Diagrama de Instância de fatos para parte do vocabulário EU-Rent.

O equivalente XML Schema deste modelo UML/MOF para o vocabulário SBVR EU-Rent de locação de automóveis pode ser representado pelo trecho da Tabela 3-6.

Tabela 3.6 - XML Schema do modelo UML/MOF para o vocabulário SBVR EU-Rent.

```
<xsd:complexType name="is-branch" >
        <xsd:complexContent >
                <xsd:extension base="esbvr:Instantiation">
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element name="branch" type="esbvr:Referent"/>
                        </xsd:choice>
                        <xsd:attribute name="branch" type="xsd:IDREF" use="optional"/>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
<xsd:element name="is-branch" type="is-branch"/>

<xsd:complexType name="branch-has-name" >
        <xsd:complexContent >
                <xsd:extension base="esbvr:Fact">
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element name="branch" type="esbvr:Referent"/>
```

```
                                        <xsd:element name="name" type="esbvr:Referent"/>
                                </xsd:choice>
                                <xsd:attribute name="branch" type="xsd:IDREF" use="optional"/>
                                <xsd:attribute name="name" type="xsd:IDREF" use="optional"/>
                        </xsd:extension>
                </xsd:complexContent>
</xsd:complexType>
<xsd:element name="branch-has-name" type="branch-has-name"/>

<xsd:complexType name="branch-owns-car" >
        <xsd:complexContent >
                <xsd:extension base="esbvr:Fact">
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element name="branch" type="esbvr:Referent"/>
                                <xsd:element name="car" type="esbvr:Referent"/>
                        </xsd:choice>
                        <xsd:attribute name="branch" type="xsd:IDREF" use="optional"/>
                        <xsd:attribute name="car" type="xsd:IDREF" use="optional"/>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
<xsd:element name="branch-owns-car" type="branch-owns-car"/>

<xsd:complexType name="is-car" >
        <xsd:complexContent >
                <xsd:extension base="esbvr:Instantiation">
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element name="car" type="esbvr:Referent"/>
                        </xsd:choice>
                        <xsd:attribute name="car" type="xsd:IDREF" use="optional"/>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
<xsd:element name="is-car" type="is-car"/>

<xsd:complexType name="car-has-vehicle-identification-number" >
        <xsd:complexContent >
                <xsd:extension base="esbvr:Fact">
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                            <xsd:element name="car" type="esbvr:Referent"/>
                            <xsd:element name="vehicle-identification-number" type="esbvr:Referent"/>
                        </xsd:choice>
                        <xsd:attribute name="car" type="xsd:IDREF" use="optional"/>
                        <xsd:attribute name="vehicle-identification-number" type="xsd:IDREF"
                       use="optional"/>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
<xsd:element name="car-has-vehicle-identification-number"
                                        type="car-has-vehicle-identification-number"/>

<xsd:complexType name="is-name" >
        <xsd:complexContent >
                <xsd:extension base="esbvr:Instantiation">
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element name="name" type="esbvr:Referent"/>
                        </xsd:choice>
                        <xsd:attribute name="name" type="xsd:IDREF" use="optional"/>
                </xsd:extension>
        </xsd:complexContent>
```

```
</xsd:complexType>
<xsd:element name="is-name" type="is-name"/>

<xsd:complexType name="is-regional-headquarters" >
        <xsd:complexContent >
                <xsd:extension base="esbvr:Instantiation">
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element name="regional-headquarters" type="esbvr:Referent"/>
                        </xsd:choice>
                        <xsd:attribute name="regional-headquarters" type="xsd:IDREF" use="optional"/>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
<xsd:element name="is-regional-headquarters" type="is-regional-headquarters"/>

<xsd:complexType name="is-vehicle-identification-number" >
        <xsd:complexContent >
                <xsd:extension base="esbvr:Instantiation">
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                             <xsd:element name="vehicle-identification-number" type="esbvr:Referent"/>
                        </xsd:choice>
                        <xsd:attribute name="vehicle-identification-number" type="xsd:IDREF"
                        use="optional"/>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
<xsd:element name="is-vehicle-identification-number" type="is-vehicle-identification-number"/>
```

O trecho da Tabela 3.7 mostra um exemplo de como os fatos do diagrama anterior podem ser representados e validados pelo XML Schema produzido a partir do modelo MOF, que por sua vez foi baseado no metamodelo SBVR.

Tabela 3.7 - Exemplo de fatos validados pelo XML Schema.

```
car-has-vehicle-identification-number car="c" vehicle-identification-number="v"/>
<branch-has-name branch="b" name="n"/>
<branch-owns-car branch="b" car="c"/>
<is-regional-headquarters regional-headquarters="b"/>
<esbvr:Thing xmi:id="b"/>
<esbvr:Thing xmi:id="c"/>
<esbvr:Text xmi:id="v">ABC123</esbvr:Text>
<esbvr:Text xmi:id="n">London Downtown</esbvr:Text>
```

3.6 Padrões e Tecnologias em Processos de Negócio na Web

A importância dada aos processos de negócio, nos últimos anos, aliada a um maior número de fornecedores e a alta complexidade das demandas de grandes clientes, impulsionaram a criação de diversos padrões e tecnologias para organizar a realização dos negócios na Web. Quando se fala em negócios via Web, os Serviços Web configuram-se

como os elementos básicos para esta realização. Os esforços realizados até agora sintetizam-se na necessidade de descrever um processo de negócio em um formato padronizado e inteligível tanto por analistas de negócios quanto por sistemas computacionais. Entretanto, o surgimento de diversos padrões diferentes, desenvolvidos por diferentes organizações, tem gerado dúvidas constantes sobre quais padrões adotar, e qual a aplicabilidade e uso de cada um deles.

Um grande esforço está sendo despendido pela OMG no sentido de criar padrões de metamodelagem de processos de negócio. A Figura 3-11 mostra alguns dos metamodelos derivados a partir do UML/MOF com influência de WS-BPEL, como linguagem de composição de serviços Web..

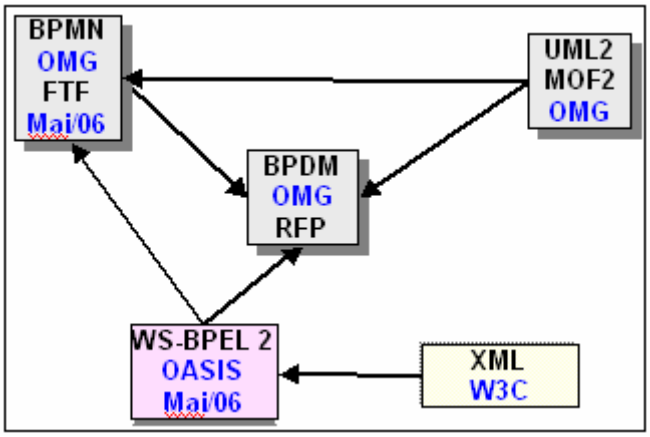

Figura 3.11 - Padrões e Metamodelos em Processos de Negócio.

As subseções a seguir apresentam alguns detalhes relacionados a esses metamodelos e dos relacionamentos entre eles.

3.6.1 Serviços Web

A tecnologia de Serviços Web provê um mecanismo de comunicação, altamente interoperável, entre os computadores de uma rede. Assim, a idéia é que, para fazer a integração de processos de negócio com outras organizações, no caso de fusão, aquisição

ou parceria não seja necessário um investimento muito grande em desenvolvimento de software para integrar os diferentes sistemas. Esta alta interoperabilidade, obtida através de um conjunto de protocolos padronizados, refere-se à capacidade de interação entre diferentes serviços Web, independentemente da plataforma de execução, da tecnologia de desenvolvimento, da linguagem de implementação, do ambiente operacional e da localização.

O sucesso crescente dos Serviços Web deve-se a vários fatores, entre os quais estes que se seguem.

- Os sistemas podem interagir dinamicamente um com o outro usando tecnologias padrões da Internet.
- Os serviços Web, uma vez construídos, podem ser reusados muitas vezes, na forma em que se encontram, sem necessidade de recompilação.
- Os Serviços Web podem ser implementados em qualquer linguagem de programação.
- Os clientes do serviço Web não precisam se preocupar com *firewalls* porque a comunicação é baseada no protocolo HTTP [100].
- Os sistemas podem anunciar suas capacidades para que outros sistemas possam usá-las automaticamente. Por exemplo, uma loja virtual pode anunciar Serviços Web para acessar dados de catálogo, para administrar o carrinho de compra e para finalizar o processo de compra.
- Outros padrões para tratar de aspectos de segurança, qualidade, confiabilidade, descoberta automática e composição estão sendo elaborados.

3.6.2 A Linguagem WS-BPEL

A linguagem WS-BPEL (Web Services – Business Process Execution Language) [32] surgiu ocupando o espaço que outras duas linguagens de execução de processos não conseguiram ocupar. A WfMC (Workflow Management Coalition) [101], que, desde 1993, vem promovendo o segmento de Workflow, patrocinou o XPDL [102] um dos padrões mais antigos nesta área. XPDL contempla alguns conceitos interessantes que as demais

especificações não exploraram adequadamente, tais como as idéias relacionadas a tarefas eminentemente humanas. A sua grande fraqueza é a obsolescência do seu modelo que não contempla conceitos e usos de Serviços Web [103]. Como uma dissidência da WfMC surgiu uma nova organização denominada BPMI [62] que, em 2002, criou o BPML [104]. Assim como o XPDL, a especificação BPML utiliza o padrão XML para descrever seus processos. Se por um lado, BPML teve o mérito de consolidar a idéia de orquestração de processos em um nível totalmente genérico, por outro não incorporou algumas das inovações da WfMC consideradas muito boas. Para a maioria das empresas o BPML estava muito distante da realidade e, portanto, não conseguiam enxergar aplicação prática.

Enquanto ocorria a polêmica entre XPDL e BPML, as empresas Microsoft, IBM, Siebel, BEA e SAP uniram-se e propuseram um novo modelo, denominado BPEL (*Business Process Execution Language*). O BPEL surgiu da união de outros padrões proprietários, mais especificamente o XLANG da Microsoft e o WSFL da IBM. Depois de concluído seu meta-modelo e especificação inicial, passaram o controle do padrão para a organização internacional OASIS, rebatizado de WS-BPEL, que encontra-se atualmente na sua versão 2.0.

Similarmente ao BPML, o WS-BPEL é totalmente aderente à especificação de serviços Web, conduzindo a sistemas de orquestração de redes de serviços na WEB. Portanto, WS-BPEL baseia-se em descrições contidas em arquivos WSDL [105]. Rapidamente, esta nova especificação começou a chamar a atenção dos principais fornecedores de solução no mercado de gerenciamento de processos baseados em serviços Web. Com o amparo de grandes empresas de TI, WS-BPEL tornou-se o padrão mais largamente aceito em curto espaço de tempo.

Como ponto negativo, o WS-BPEL, apesar de apresentar muitas provas de conceito, ainda não há muitas aplicações práticas. Assim como o BPML, o WS-BPEL não possui uma diferenciação para tarefas humanas, o que é explicável pelo paradigma que adota, mas um ponto que aumenta o nível de complexidade no seu uso.

Neste cenário, é grande a probabilidade de que a especificação de WS-BPEL seja enriquecida com muitas das boas idéias de outros padrões.

3.6.3 O Metamodelo BPMN

Ao contrário das especificações anteriores, o BPMN (Business Process Modeling Notation) [106] é uma especificação para modelagem visual de processos de negócio. Visa proporcionar às organizações a capacidade de entender seus procedimentos de negócio internos em uma notação gráfica e que dê a elas a habilidade para comunicar estes procedimentos de uma maneira padronizada. O metamodelo BPMN tem como objetivo prover uma interface simples e poderosa que seja compreensível para os analistas de negócio que criam os modelos iniciais dos processos; para os desenvolvedores responsáveis por implementar estes processos; e também para as pessoas de negócio que administrarão e monitorarão esses processos. BPMN provê também um modelo para a geração de código em WS-BPEL. Assim, é possível criar uma ponte entre um modelo de processo de negócio descrito em BPMN e a implementação deste processo.

BPMN define um Diagrama de Processo de Negócio (BPD – *Business Process Diagram*) baseado em fluxogramas para criar modelos gráficos de operações de processos de negócio. Um BPD, então, é uma rede de elementos gráficos que são as atividades e os controles de fluxo que definem a ordem em que eles serão executados. Os BPDs representam diagramas simples bastante familiares à maioria dos analistas de negócio, tal como o diagrama de fluxos da Figura 3.12, extraído de [106]. Por exemplo, atividades são retângulos e decisões são losangos. Deve ser enfatizado que, BPMN, embora facilite a criação de diagramas simples para os modelos de processo de negócio, ao mesmo tempo apresenta mecanismos para controlar a complexidade inerente a processos de negócio. As quatro categorias básicas de elementos considerados em BPMN são: Objeto de Fluxo, Objetos de Conexão, *Swimlanes* e Artefatos. Detalhes sobre estes elementos podem ser encontrados na especificação BPMN [106].

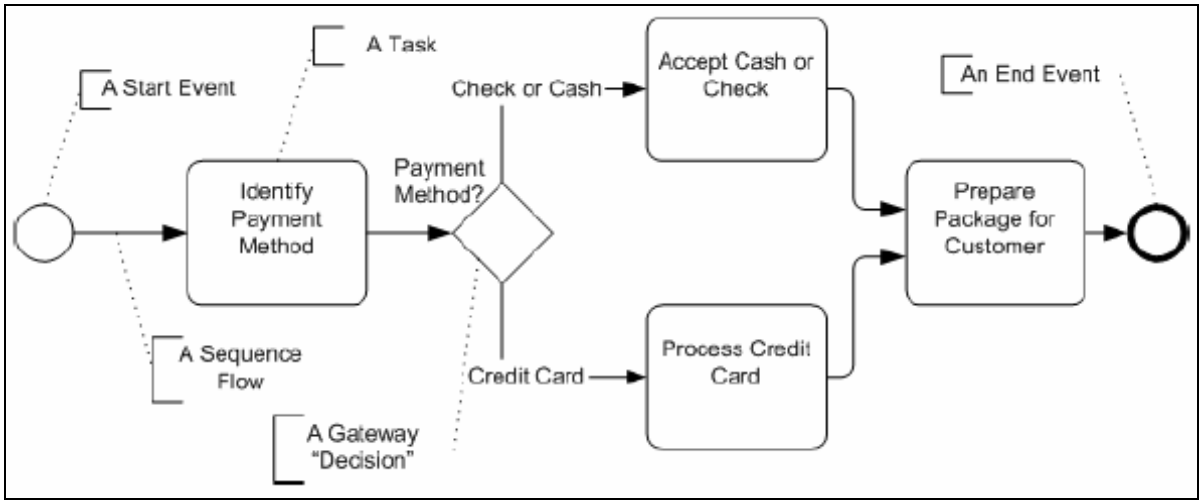

Figura 3.12 - Exemplo de um diagrama de processo em BPMN.

BPMN consolida as melhores idéias de outros esforços de padronização, tais como as anotações ou metodologias oriundas dos diagramas de atividades UML [107], diagramas de processos de negócio do perfil UML/EDOC [108], diagramas de atividades da notação IDEF [109], diagramas de processos de negócio do ebXML BPSS [110] e diagramas de fluxo de atividade ADF do WebSphere/IBM [111].

Entre os padrões atuais que provêem notações gráficas para gerenciamento de processos, o BPMN parece ser a única unanimidade, pois tem apoio de todos os grandes fornecedores de solução e não possui nenhuma outra especificação concorrente.

3.6.4 O Metamodelo BPDM

BPDM - *Business Process Definition Metamodel* é um metamodelo que define um conjunto de elementos abstratos de processos de negócio para especificação de processos de negócio executáveis e as interações entre eles, entre diferentes organizações, num ambiente distribuído. Os modelos resultantes devem ser independentes de plataforma de execução, ou seja, devem estar no nível PIM do MDA. O metamodelo propõe tratar não somente dos processos computacionais, mas também dos processos manuais e prevê um esquema baseado em XMI para realizar o intercâmbio de modelos. Isto projeta o BPDM como um

metamodelo para realizar a ponte entre vários outros metamodelos e linguagens, tais como para BPMN, WS-BPEL e UML, colaborando, assim, para a agilidade necessária aos negócios da atualidade.

BPMN e BPDM são padrões projetados para uso por pessoas de negócio, para representar modelos de negócio ao invés de modelos de execução como o fazem WS-BPEL ou WS-CDL [112]. BPMN é um padrão para a representação gráfica de processos de negócio, enquanto BPDM é um padrão para arquivos XML nos quais a representação gráfica pode ser armazenada. Assim, uma ferramenta poderia criar um processo de negócio em notação BPMN e poderia salvar esta notação como modelos BPDM.

Na medida em que BPDM está ainda numa fase de definição muito preliminar, há ainda muita discussão sobre como capturar os aspectos de orquestração e coreografia de processos e representá-los equivalentemente em modelos gráficos de BPMN.

3.6.5 Mapeamento de Modelos no Contexto de MDA

A Figura 3.13 adaptada a partir da especificação OMG-PRR [84], mostra como os metamodelos posicionam-se na arquitetura orientada a modelos (MDA – *Model Driven Architecture*).

A Figura 3.14 também adaptada a partir de OMG-PRR [84], mostra como poderia ocorrer o mapeamento entre os padrões. Conforme pode ser visto, os metamodelos que estão no nível CIM de MDA, tais como BPMN, SBVR e BMM, modelam aspectos de negócio de maneira independente de tecnologias de processamento, nos quais, os agentes de processamento podem ser humanos, organizações, ou sistemas computacionais. Neste contexto, a portabilidade de modelos será conduzida através de outros metamodelos, tais como BPDM, PRR e ODM, que proverão mecanismos de transformação e serialização para o intercâmbio de modelos. Por exemplo, a partir de um modelo no nível CIM é possível mapear para modelos nos níveis PIM/PSM que podem incluir mapeamentos para máquinas BPEL ou mecanismos de regras que executem descrições em SWRL, ou que implementem a especificação JSR 94, ou quaisquer outros mecanismos proprietários.

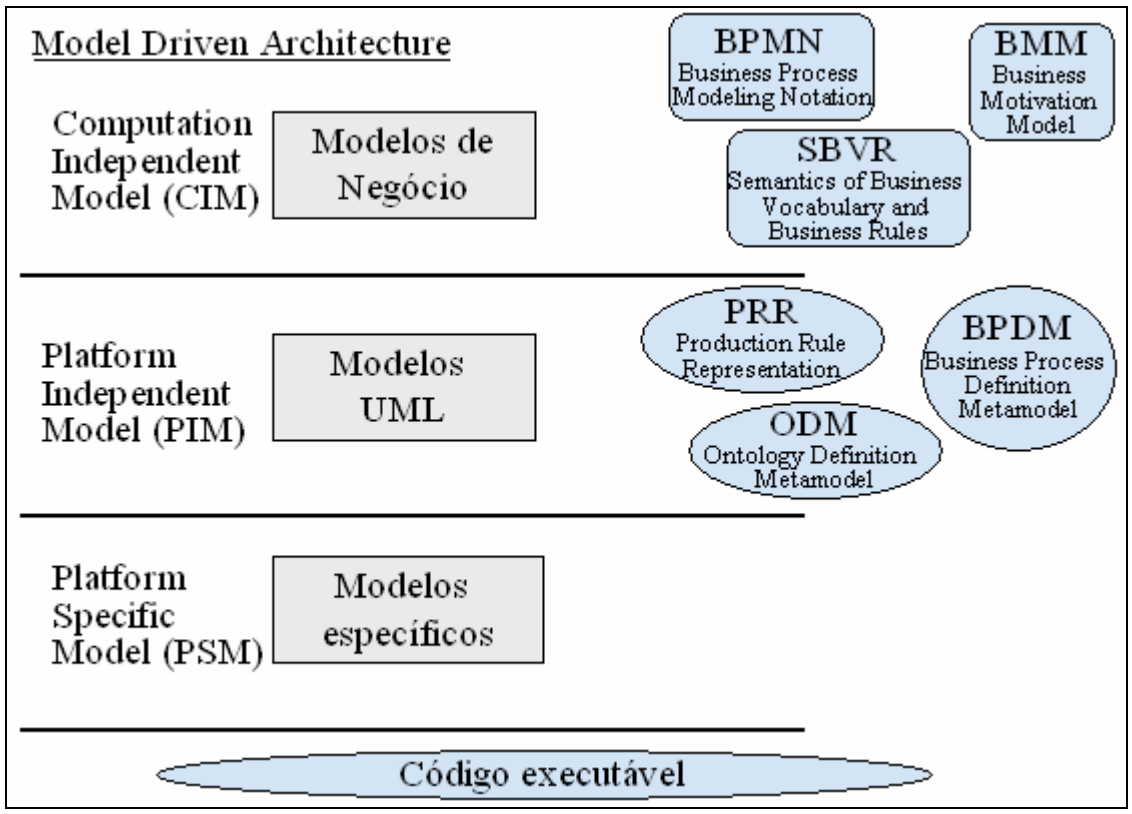

Figura 3.13 - Metamodelos no contexto de MDA.

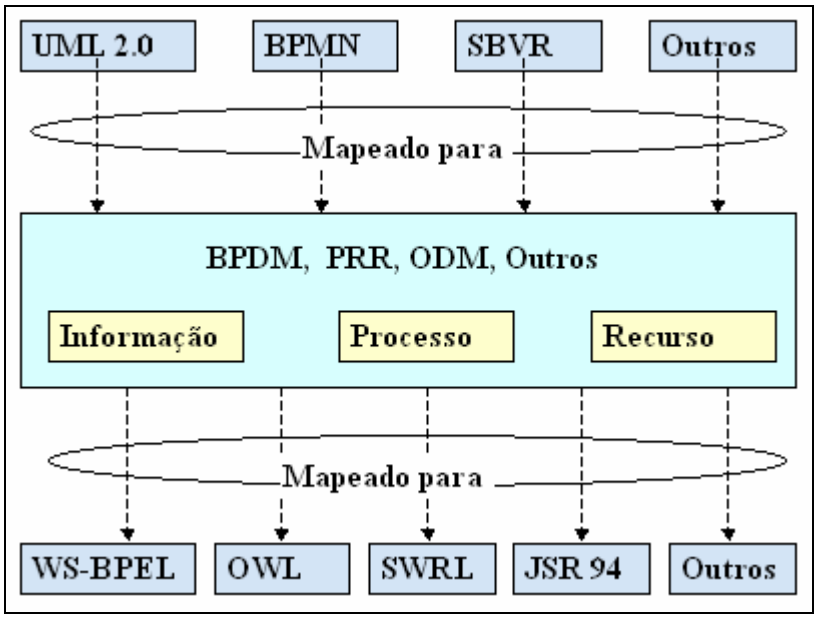

Figura 3.14 - Exemplo de mapeamento entre padrões.

3.7 Considerações Finais do Capítulo 3

Analisando-se o histórico e os movimentos dos principais fornecedores neste mercado de gerenciamento de regras de negócio e de processos de negócio chega-se à conclusão de que ambas as áreas precisam convergir na busca de uma solução integradora que torne mais flexível a criação e manutenção de sistemas computacionais. É necessário que as regras de negócio sejam externalizadas para facilitar o seu gerenciamento por analistas de negócio e, ao mesmo tempo, garantir que estas regras de negócio sejam executadas de maneira integrada com os processos de negócio. Por um lado, conforme SOA vai ganhando adeptos nas organizações, o WS-BPEL vai se transformando na principal especificação técnica para gerenciamento de processos de negócio, o BPMN torna-se uma boa opção para a modelagem de fluxos de processos, que, por ser gráfica, facilita o entendimento e uso para as pessoas de negócio. Por outro lado, com a volta do interesse pelas regras de negócio, surgiram novos mecanismos de regras de negócio e novos metamodelos para tratamento de regras, o SBVR promete ser uma boa alternativa para o desenvolvimento de ferramentas ou ambientes de modelagem de regras de negócio que incorporem os termos, fatos e regras na linguagem das pessoas de negócio.

Assim, o desenvolvimento e a manutenção de sistemas computacionais de maneira mais flexível e rápida passa pela construção de ferramentas ou ambientes de desenvolvimento de software que combinem os modelos e as tecnologias para gerenciamento de regras de negócio e gerenciamento de processos, considerando as recomendações propostas pela arquitetura MDA.

Capítulo 4

O Modelo de Serviços baseado em Regras de Negócio

O propósito principal deste trabalho consiste em apresentar conceitos para desenvolvimento de regras e execução de serviços, baseados em regras de negócio. São propostas uma plataforma de desenvolvimento de regras e uma plataforma de execução que permitem refletir rapidamente nos sistemas computacionais as mudanças demandadas no dia-a-dia dos negócios das empresas e dos governos. Este capítulo detalha a arquitetura proposta como uma evolução dos trabalhos propostos em [113, 114]. Na seção 4.1 são descritos os módulos que compõem a arquitetura e na seção 4.2 detalhados os repositórios de modelos e metadados que suportam a abordagem proposta. Na seção 4.3 apresentam-se alguns exercícios de transformação de regras descritas na terminologia das pessoas de negócio para regras computacionalmente executáveis.

4.1 Os Módulos da Arquitetura

A Figura 4-1 mostra a Plataforma de Execução de Serviços. Os módulos e repositórios mais importantes são:

- Preparador de Serviço
- Executor de Regras de Negócio
- Mecanismo de Registro de Serviços.
- Ambiente de Desenvolvimento de Regras de Negócio e Ontologias
- Repositório de Vocabulários e Regras de Negócio
- Repositório de Perfis de Serviços

- Repositório de Ontologia de Serviços

A parte inferior da Figura 4.1 mostra a Plataforma de Desenvolvimento com o Ambiente Integrado de Desenvolvimento de Regras e Ontologias, que faz a criação e manutenção de vocabulários, regras e ontologias de serviço nos repositórios. Esses repositórios são usados concorrentemente pela Plataforma de Execução de Serviços. O Ambiente de Desenvolvimento de Regras e Ontologias tem a ajuda de análises e otimizações feitas a partir de históricos de execução gerados pelo Executor.

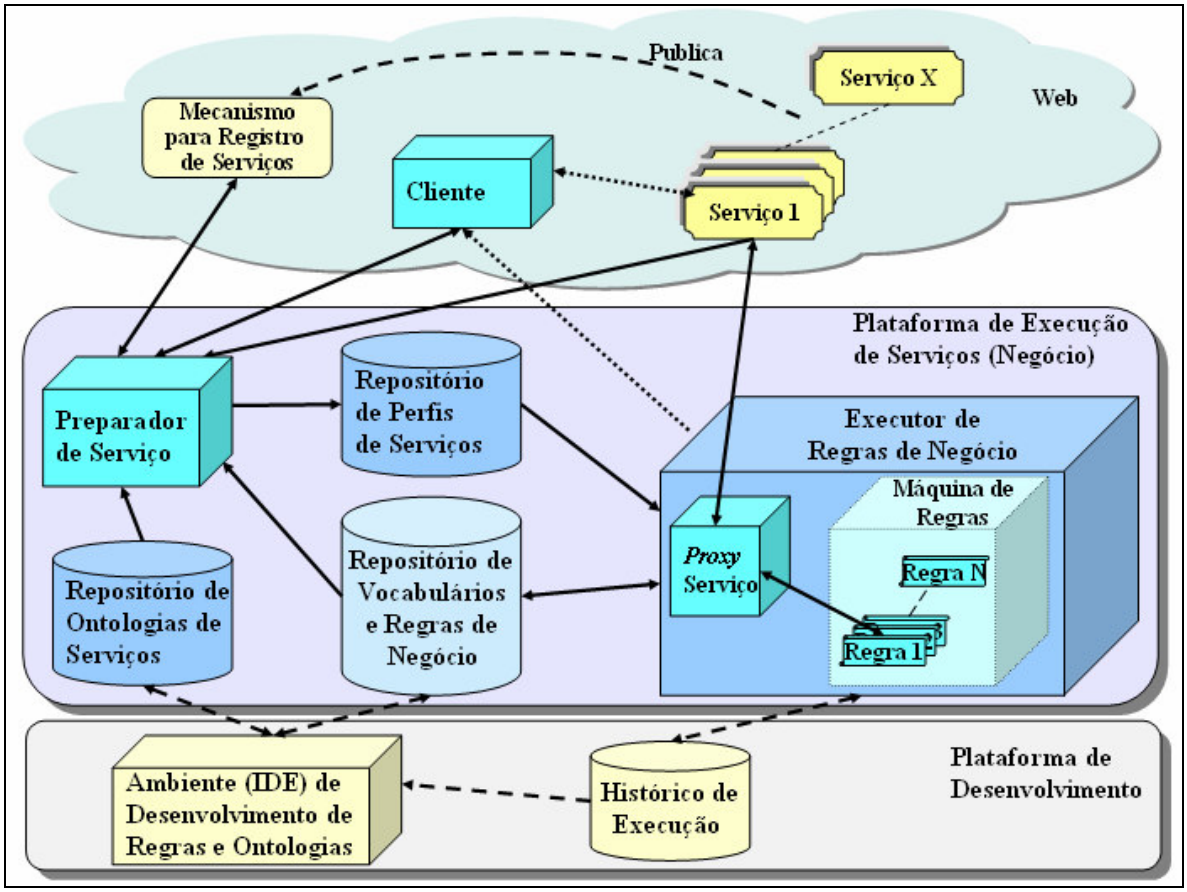

Figura 4.1 - Arquitetura de Execução de Regras de Negócio.

As citações aos módulos mais referenciados da arquitetura serão, doravante, abreviados. Assim, o Preparador de Serviço será chamado de **Preparador**, o Executor de Regras de Negócio de **Executor**, o Repositório de Vocabulário e Regras de **Repositório de Regras**, o Repositório de Ontologias de Serviços de **Repositório de Ontologias**, o Mecanismo de

Registro de Serviços de **Mecanismo de Registro** e o Ambiente Integrado de Desenvolvimento de Regras e Ontologias de **IDE** (*Integrated Development Environment).*

O diagrama da Figura 4.2 ilustra a seqüência de ações executadas pelos módulos e repositórios da plataforma de execução proposta.

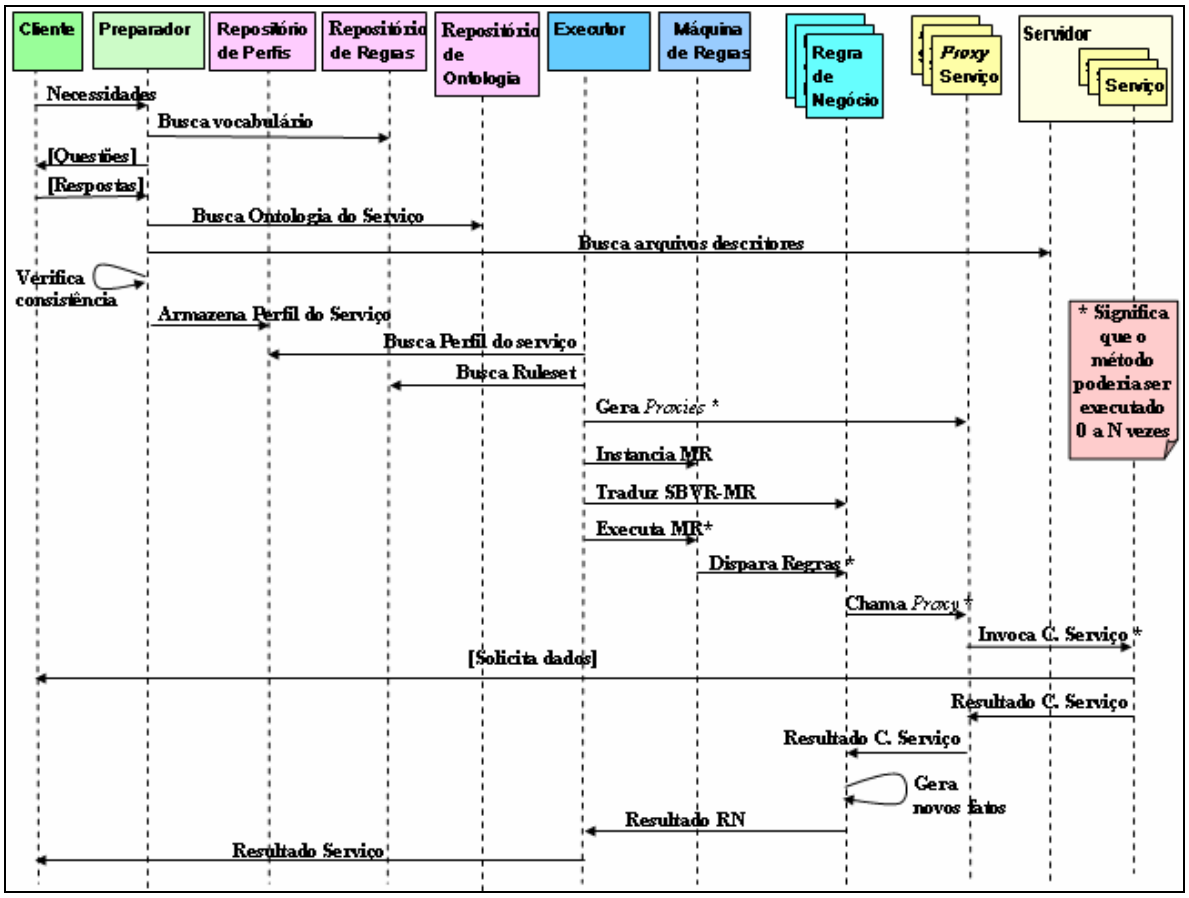

Figura 4.2 - Arquitetura - Diagrama de Seqüência.

O módulo Cliente interage com o Preparador para descobrir uma necessidade que, na prática, conduz à realização de um serviço. O Preparador obtém do Repositório de Regras e do Repositório de Ontologia os elementos relacionados com a descrição do Cliente e, se for o caso, interage novamente para determinar a necessidade. Estes elementos são consolidados numa entidade denominada de Perfil de Serviço que será armazenada no Repositório de Perfis. O Executor é responsável por recuperar no Repositório de Regras as regras de negócio pertinentes, instanciar uma máquina de regras, traduzir as regras de negócio, recuperar os localizadores (também referidos como, *interaction point*, *endpoint* ou

URL – *Universal Resource Locator*) dos componentes de serviços[1] a partir do Mecanismo de Registro, gerar os *proxies* para os componentes de serviço, executar as regras de negócio e devolver o resultado ao Cliente. Os elementos dos repositórios são criados e gerenciados pelo IDE.

As próximas sub-seções apresentam mais detalhes dos principais módulos desta arquitetura.

4.1.1 Ambiente de Desenvolvimento de Regras de Negócio e Ontologias

A arquitetura pressupõe o uso de um IDE para definir e gerenciar os elementos (termos, verbos, qualificadores e questões que compõem o vocabulário e as regras de negócio) que são usados por uma comunidade de negócio. Estes elementos devem estar precisamente definidos e estruturados nos repositórios, de maneira que o acesso seja fácil e consistente. O IDE acessa o Repositório de Regras e o Repositório de Ontologias para atualizar as regras de negócio e as ontologias, ao mesmo tempo que o Executor consulta estes repositórios e busca as regras para executar. Assim, um aspecto importante no IDE é que ele se apóia no uso de ontologias para estruturar os serviços, levando em consideração as regras de negócio contidas no Repositório de Regras, os serviços publicados no Mecanismo de Registro e o histórico de execução das regras de negócio.

O componente mais importante neste IDE é o editor. O editor provê facilidades para definir e manter os elementos que compõem os vocabulários, as regras de negócio e as ontologias. O editor, aqui proposto, baseia-se no uso de modelos (*templates*) associados aos diferentes tipos de regras existentes. O metamodelo SBVR define alguns tipos de regras, tais como aqueles que tratam de obrigações, proibições e necessidades. Aproveitando as idéias de Ross [115], que propõe o uso de modelos para facilitar a especificação de regras de negócio, este editor provê um conjunto de modelos, associados aos tipos de regras definidos em SBVR. Assim, para o tipo modal "*é obrigatório que"* que trata de obrigações o modelo tem a seguinte sintaxe**: "É obrigatório** [**que**] <fato> [**se/enquanto** <condição>]".

[1] ou *software service*, conforme denominação dada pela OMG no pedido de propostas para o metamodelo SSM – *Software Service Metamodel*.

Como alternativa o editor provê um modelo equivalente, conforme sugerido no Anexo C de SBVR, onde a obrigatoriedade é representada com a inclusão da palavra-chave "**deve"** no corpo do <fato> e excluindo-se a palavra-chave do tipo modal "É obrigatório que". Desta forma, a sintaxe para este modelo equivalente tem o seguinte formato: "... **deve** ... [**se/enquanto** <condição>]". Portanto, o IDE de Regras inclui também uma estrutura interna para armazenar os modelos e um conjunto de operações para a manutenção dos modelos.

Embora nem sempre de percepção imediata, alguns fatos que definem o tipo da regra representam ações, cujas realizações estão associadas à execução de parte do serviço. Assim, alguns <fato>s que compõem o tipo modal "*é obrigatório que"* traduzem-se em ações, que, uma vez realizadas, terão como resultado a asserção de que os fatos são verdadeiros. Ações como esta, de modo geral, representam segmentos da lógica de negócio já bastante estabilizados, no sentido de pouca vulnerabilidade a mudanças, de modo que o encapsulamento destas ações como componentes de serviço torna-se bastante interessante. Sendo assim, as próprias ações ou os próprios nomes das regras dão indícios da necessidade ou não de um componente de serviço para realizar tal ação. Portanto, no instante de criação ou edição de uma regra de negócio o Editor pesquisa o Mecanismo de Registro por tais componentes de serviço e faz uma "ligação a priori" entre a regra de negócio e os componentes de serviço associados às ações. A ligação antecipada é bastante providencial porque evita-se o risco de, no instante de execução, não se encontrar um determinado componente de serviço que realizaria parte do serviço. Neste instante, o Editor agrega informações desta ligação antecipada num estrutura denominada Ontologia, associada ao serviço. Esta Ontologia de Serviço é armazenada no Repositório de Ontologia para que, no instante de execução o Executor, já de posse das descrições dos componentes de serviço, prepare as invocações.

O diagrama de casos de uso da Figura 4.3 mostra os principais processos e funcionalidades deste Editor.

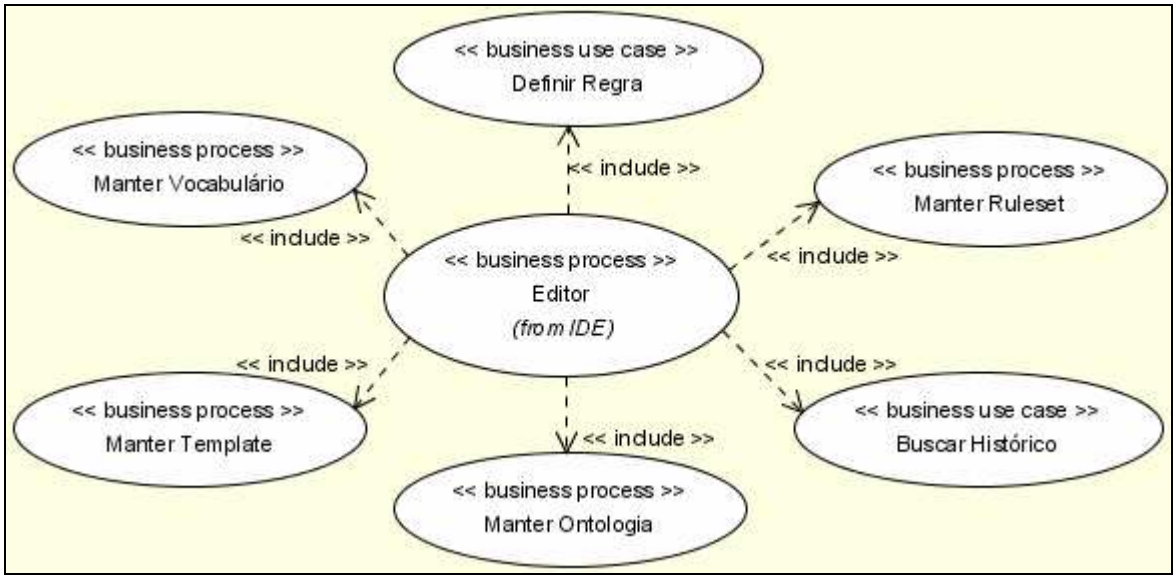

Figura 4.3 - Casos de Usos para o Editor.

O Editor contempla as operações básicas recomendadas por alguns padrões de projeto (*design pattern*) para buscar (*get*), atualizar (*set*), incluir e excluir todos os elementos sob seu controle. Desta forma, os processos (estereotipados no diagrama como <<business process>>) "Manter *Ruleset*", "Manter Vocabulário", "Manter Template" e "Manter Ontologia" contemplam estas operações básicas, com pequenas diferenças. A Figura 4.4 com o diagrama para o processo "Manter Ruleset" ilustra estas operações básicas sobre regras e *rulesets.*

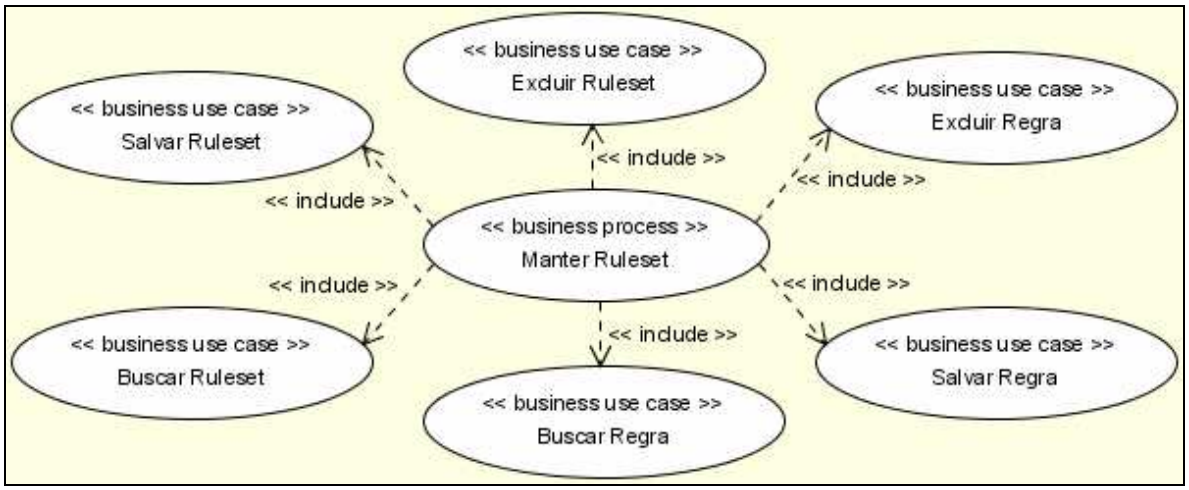

Figura 4.4 - Casos de Uso para Manter *Ruleset*.

Além destas operações básicas o Editor contempla as operações específicas para cada um dos processos que tratam os elementos. Por exemplo, o diagrama para o processo "Manter Vocabulário" mostrado na Figura 4.5 inclui a operação "Estruturar Vocabulário" no Repositório de Regras. Esta operação consiste na instanciação do modelo de vocabulário do Repositório de Regras, mostrado na Figura 4.12.

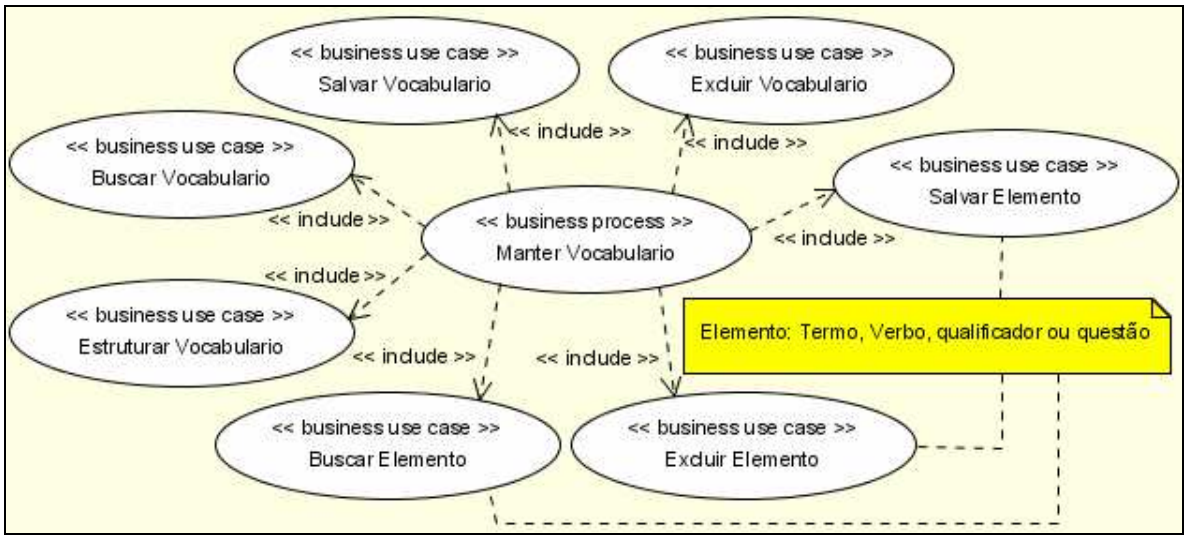

Figura 4.5 - Casos de Uso para Manter Vocabulário.

A definição de uma regra, representada pelo caso de uso "Definir Regra" no processo "Editor", envolve várias outras operações, conforme mostra a Figura 4.6.

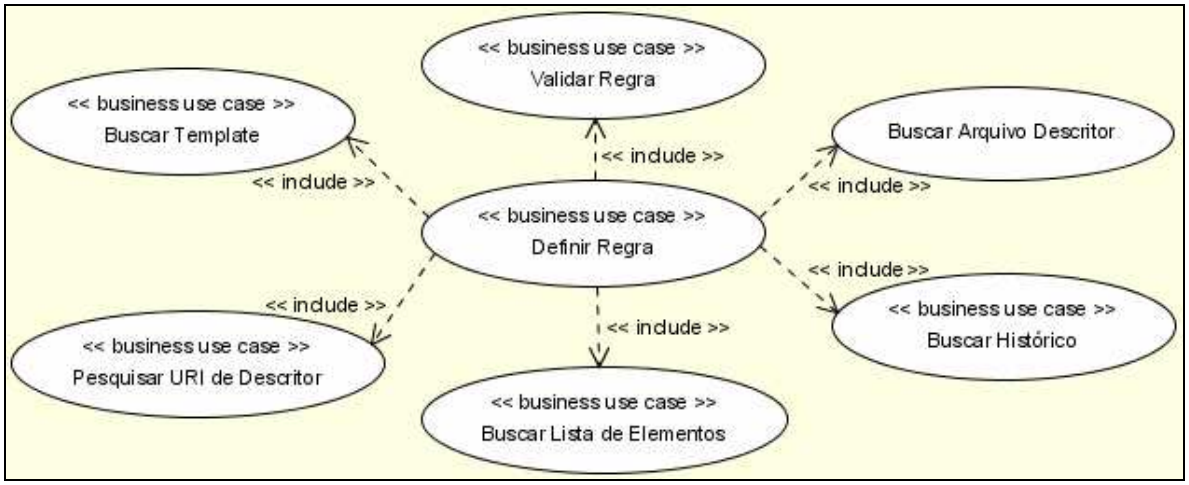

Figura 4.6 - Casos de Uso para Definir Regra.

Assim, para “Definir Regra”

- O analista de negócio seleciona no Editor o vocabulário da sua comunidade.
- O analista de negócio seleciona o modelo desejado.
- O Editor executa “Buscar Template” e apresenta-o numa janela do Editor.
- Quando o analista de negócio aponta para um campo do modelo o Editor executa “Buscar Lista de Elementos” para buscar a lista associada ao campo. Por exemplo, no modelo **“É obrigatório** [**que**] <fato> [**se/enquanto** <condição>]”, apontando-se para <fato> ou <condição> busca-se a lista de predicados relativos ao vocabulário selecionado. Uma vez escolhido o predicado e apontando-se para um de seus campos busca-se no vocabulário a lista de termos que fazem sentido naquele contexto.
- Alguns <fatos> e <condições>, para se tornarem verdadeiros, pressupõem a realização de alguma ação, que pode ser realizada por um componente de serviço. Com as palavras contidas num <fato> ou <condição> o Editor executa “Pesquisar URI de Descritor” para descobrir se há algum componente que realiza a ação.
- Para cada URI de Descritor válido o Editor executa “Buscar Arquivo Descritor”.
- O Editor abre cada arquivo descritor e obtém os elementos-chave, tais como, o URI do componente de serviço, os nomes das operações e os nomes dos respectivos parâmetros. Estes elementos são inseridos na Ontologia de Serviço à qual a regra em definição deve ser associada.
- O analista de negócio pode selecionar “Validar Regra” para verificar a sintaxe.
- Por fim, o analista de negócio seleciona “Incluir Regra”.

A Figura 4.7 mostra as associações da classe do processo Editor com as várias classes que representam os elementos tratados por ele.

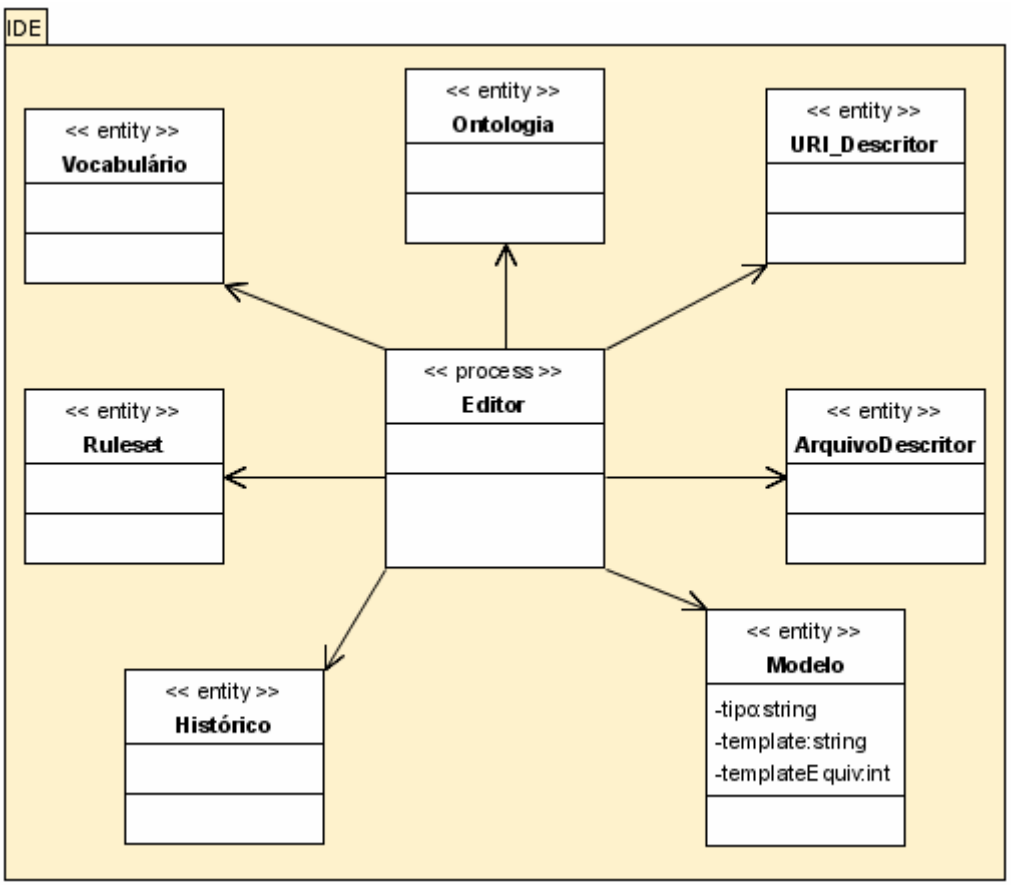

Figura 4.7 - Diagrama de Classes do Editor.

4.1.2 O Cliente

O Cliente é simplesmente uma interface para o Preparador, que detém toda a lógica de interação, de controle, de acesso aos repositórios e de consulta aos mecanismos de registro de serviços. O Cliente tanto pode ser uma aplicação *desktop*, uma aplicação Web, um portal Web ou outro serviço, que fornece uma descrição da sua necessidade.

4.1.3 Preparador de Serviço

O Preparador é efetivamente o módulo responsável por "descobrir" a necessidade de realização de um serviço no contexto dos negócios entre empresas, pessoas e governos. É fortemente dependente do Repositório de Regras e do Repositório de Ontologias.

O Preparador interage com o Cliente tendo como objetivo descobrir a necessidade e fazer algumas verificações sobre a exeqüibilidade do serviço que realizará a necessidade do Cliente. De modo geral, quando o Cliente é uma aplicação totalmente automatizada, ou seja, não depende de interação com pessoas, o serviço já está precisamente definido, restando apenas fazer as verificações. Como isto ocorre em boa parte dos negócios categorizados como B2B (*Business to Business*), B2G (*Business to Government*) e G2G (*Government* to *Government)* a tarefa do Preparador fica simplificada pelo menos no que se refere à descoberta do serviço.

Quando o Cliente é uma aplicação que inclui interação com pessoas o negócio é categorizado como B2C (*Business to Customer*) ou G2C (*Government to Citizen*). Nestes casos, o Preparador deve recuperar do Repositório de Regras os termos, verbos, qualificadores e questões para orientar o diálogo, conforme proposta na seção 3.3.1, com o objetivo de descobrir o serviço. No final deste diálogo, com as informações fornecidas pelo Cliente, o Preparador terá elementos para concluir qual é o serviço e qual a razão que embasa a necessidade da realização do serviço.

A Figura 4.8 mostra o diagrama com os casos de uso do Preparador.

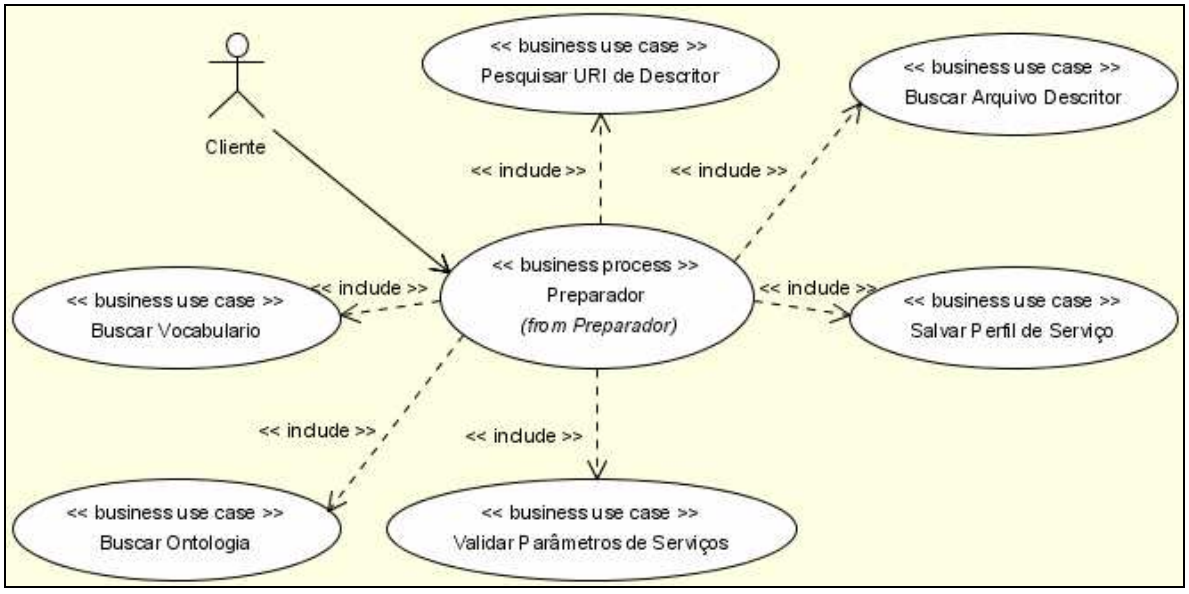

Figura 4.8 - Casos de Uso do Preparador.

A partir das descrições fornecidas pelo Cliente, ele busca através de palavras-chave (caso de uso Buscar Vocabulário) no Repositório de Regras o vocabulário (um conjunto de termos, verbos, qualificadores e questões) que tem relação direta com os termos contidos nas descrições. Caso o Preparador não consiga identificar, de forma inequívoca, todos os recursos para realizar o serviço, são feitas novas interações até que se descubra a necessidade, de forma consistente.

Uma vez definido o serviço, o Preparador faz duas verificações: (1) verifica se todos os componentes de serviço necessários para realizar o serviço estão realmente disponíveis e (2) verifica a consistência dos dados fundamentais de cada um dos componentes de serviço. Estas verificações vão garantir que a geração de código executável, a tradução de regras do modelo SBVR para a máquina de regras alvo e a conseqüente execução só serão iniciadas após certificar-se que todos os recursos necessários estarão disponíveis.

Para verificar se os componentes de serviço estão disponíveis o Preparador obtém do Repositório de Ontologia (Buscar Ontologia) os nomes ou identificadores dos componentes de serviço e usa-os para pesquisar os localizadores no Mecanismo de Registro (Pesquisar URI de Descritor). Um localizador aponta para o arquivo descritor de um componente de serviço. Com estes localizadores obtém-se os arquivos descritores dos respectivos componentes de serviço (Buscar Arquivo Descritor). Passada esta etapa pode-se concluir que os componentes de serviço estão disponíveis.

Para verificar a consistência de dados fundamentais (Validar Parâmetros de Serviços) o Preparador abre cada um dos arquivos descritores e verifica se os parâmetros de entrada do respectivo componente de serviço podem ser preenchidos com os dados disponíveis ou oriundos do Cliente. Notar que a indisponibilidade de algum dado fundamental como parâmetro de entrada de um componentes de serviço inviabiliza a realização do serviço, e, portanto, não faz sentido o início das atividades de geração de código, tradução de regras e execução das regras e dos Serviços .

Após estas verificações o Preparador determinará os elementos necessários para a realização do serviço, tais como, o identificador do serviço, o nome do serviço, os dados

oriundos do Cliente, os identificadores dos arquivos descritores de cada um dos componentes de serviço necessários, etc. Estes elementos estarão consolidados no Perfil de Serviço, que será armazenado no Repositório de Perfis (Salvar Perfil de Serviço).

A Figura 4.9 mostra as classes que colaboram para a realização do Preparador.

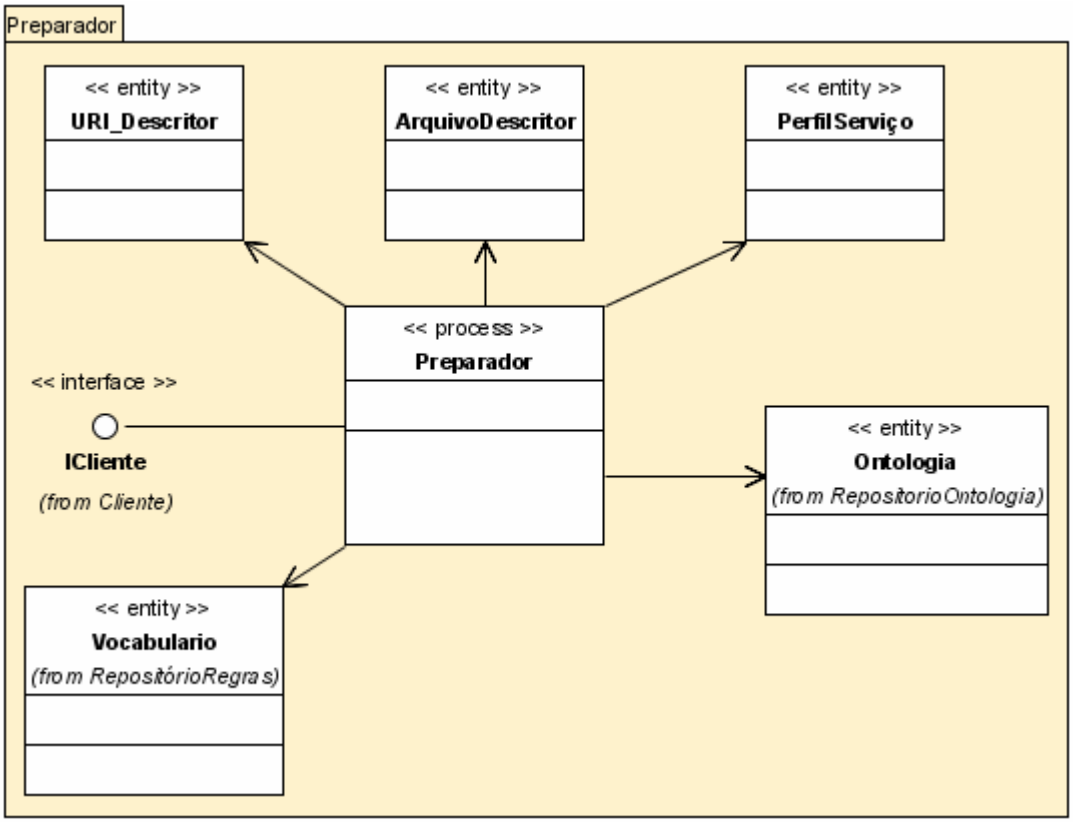

Figura 4.9 - Diagrama de Classes do Preparador.

4.1.4 Executor de Regras de Negócio

O Executor é o módulo responsável por disponibilizar os recursos, configurar os parâmetros de execução e executar as regras de negócio. A disponibilização de recursos consiste na instanciação da máquina de regras, tradução de regras da representação SBVR para modelos executáveis e geração de programas clientes para chamar os serviços remotos. A configuração de parâmetros de execução consiste no mapeamento de parâmetros obtidos do Perfil de Serviço para os recursos disponibilizados e na geração de

fatos que possam disparar a execução das regras. A execução das regras de negócio, que por sua vez chamarão a execução dos processos de negócio, realiza-se com o suporte destes recursos configurados.

O diagrama de casos de uso da Figura 4.10 mostra as principais funcionalidades do Executor.

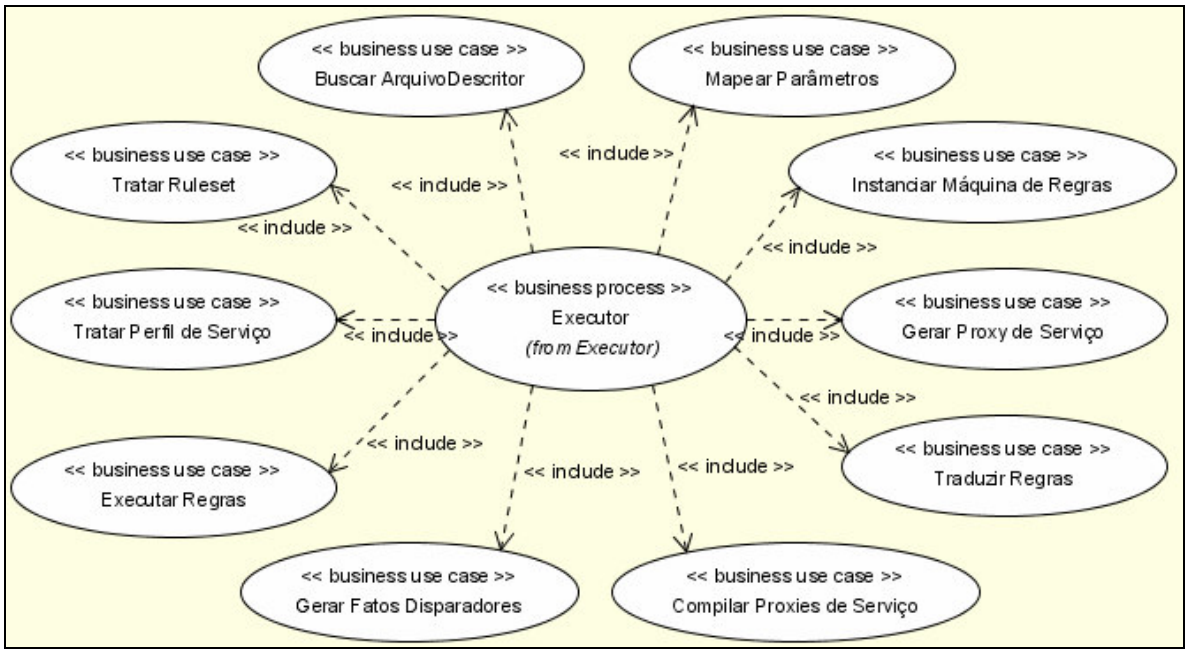

Figura 4.10 - Diagrama de Casos de Uso do Executor.

Apresenta-se a seguir uma descrição resumida de cada uma das funcionalidades representadas pelos casos de uso da Figura 4-10.

- Tratar Perfil de Serviço – Pesquisa o Repositório de Perfis periodicamente para verificar se há novos Perfis de Serviço a serem tratados. Caso positivo, para cada Perfil de Serviço extrai-se o identificador ou nome do serviço.
- Tratar Conjunto de regras – Baseado no identificador ou nome do serviço pesquisa o Repositório de Regras para obter o conjunto de regras (*ruleset*) associadas ao serviço.
- Instanciar Máquina de Regras – Cria um instância da máquina de regras escolhida para fazer inferências e executar as regras. Esta instanciação disponibiliza uma

memória de trabalho onde devem ser carregadas as regras e os fatos a serem considerados na realização do serviço.

- Gerar Proxies de Serviço – Baseado nos arquivos descritores gera os *proxies* para os componentes de serviço, ou seja, gera os programas-cliente que invocarão os respectivos componentes de serviço. Esta geração de *proxies* é feita a partir de um *proxy* genérico que é usado como modelo e os elementos-chave no arquivo descritor, tais como o *location* (*endpoint*), nome da operação e parâmetros.
- Mapear Parâmetros – Mapeia os parâmetros contidos no Perfil de Serviço para cada um dos proxies gerados.
- Traduzir Regras – A partir de um ruleset, que contém regras de negócio em notação aderente ao metamodelo SBVR traduz estas regras para o modelo da máquina de regras instanciada, já com as devidas marcações para as chamadas aos componentes de serviço. Estas regras traduzidas são armazenadas em um arquivo temporário.
- Compilar Proxies de Serviço – A partir dos proxies gerados cria um script de compilação e executa este script para compilá-los.
- Gerar Fatos Disparadores – Verificar se no Perfil de Serviço existem razões explícitas para que o serviço seja realizado. Se existirem, para cada razão, gerar um fato na memória de trabalho da máquina de regras instanciada. Estes fatos, se existirem, deverão disparar a execução das primeiras regras.
- Executar Regras – Realizar as seguintes atividades, ciclicamente, até que o serviço seja realizado, ou seja, até que ocorra a asserção verdadeira do fato associado ao serviço:
    - Executar a Máquina de Regras instanciada considerando as regras e fatos carregados na memória de trabalho. Algumas ações geram fatos diretamente na memória de trabalho, outras geram fatos após o retorno de uma chamada a um componente de serviço.
    - Verificar se algum evento ou sinal externo chegou à interface Cliente. Se sim, gerar os respectivos fatos na memória de trabalho.

O diagrama de classes da Figura 4.11 mostra as relações entre a classe do processo Executor e as classes que o suportam.

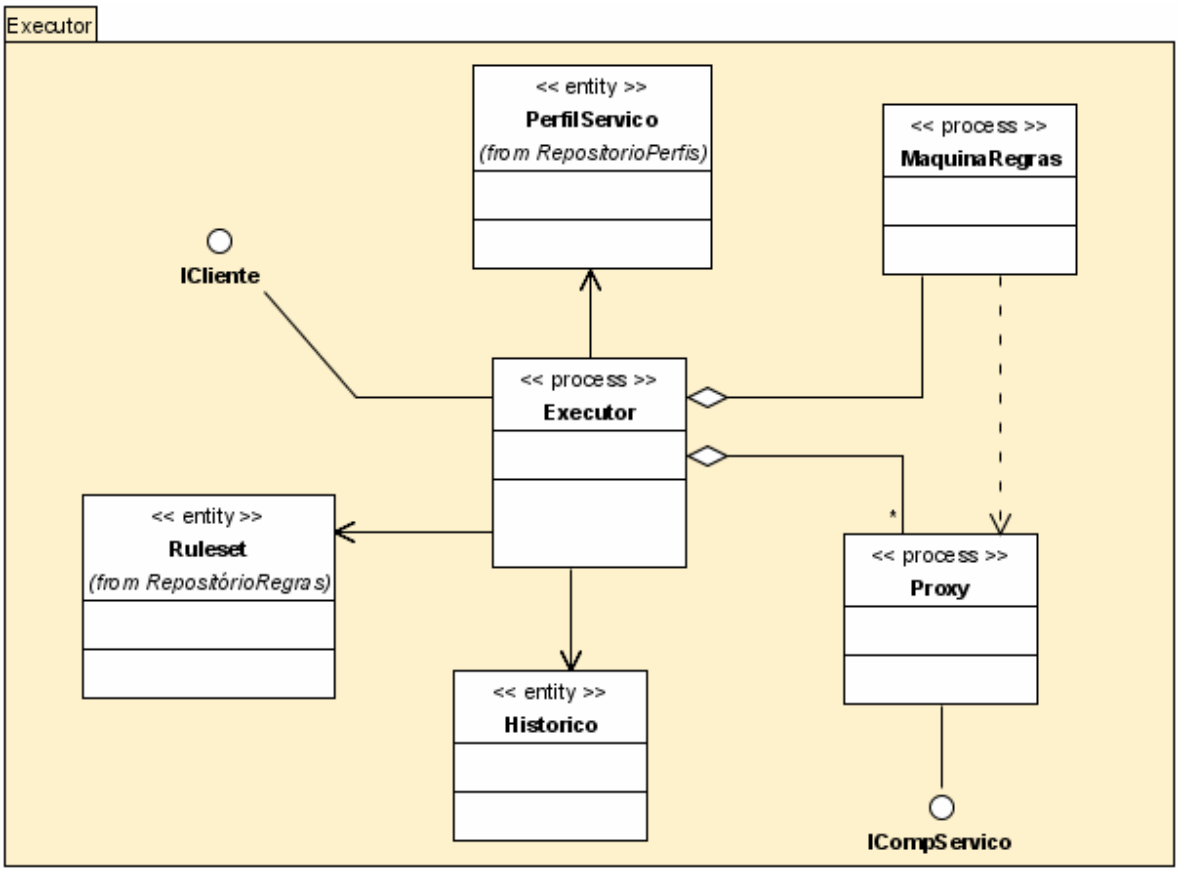

Figura 4.11 - Diagrama de Classes do Executor.

A execução da Máquina de Regras disparará a execução das regras de negócio, conforme a existência de fatos na sua memória de trabalho que "casem" com as condições contidas nas regras de negócio. A execução de uma regra de negócio pode ter como resultado a chamada de um dos *proxies* gerados, que por sua vez invocará o respectivo componente de serviço. A execução de um componente de serviço pode ter como resultado a realização de uma atividade do processo de negócio, por exemplo, o pagamento de uma taxa ou fatura, que uma vez realizada, determinará a asserção de um novo fato na memória da Máquina de Regras. Com este novo fato, novas regras de negócio poderão ser disparadas. E, assim, sucessivamente até que o serviço seja completado, novos fatos e novas regras de negócio serão executados.

4.1.5 Mecanismo de Registro de Serviços

Uma das instâncias mais populares de Mecanismo de Registro é a representada pelo UDDI (*Universal Description, Discovery, and Integration)* [116]. As facilidades providas pelo Mecanismo de Registro podem ser vistas de forma análoga às providas por uma lista telefônica. Supondo que um usuário necessite realizar uma revisão no seu automóvel e queira saber onde e como realizar o serviço, ele poderá se valer de uma lista telefônica e procurar nas páginas amarelas (classificados) pela seção "Automóveis - serviços" e terá uma lista de oficinas de todas as marcas de automóveis. Poderá filtrar um pouco mais e procurar somente as oficinas especializadas na marca do seu automóvel e localizadas na sua cidade. O eventual telefone obtido seria equivalente ao localizador do arquivo descritor, pois a partir do telefone o usuário "negociaria" um horário, preço, tempo, forma de pagamento, etc. Ou seja, a descrição de como o serviço seria realizado estaria implícita nesta "negociação" e, uma vez ambas as partes concordado com esta descrição, o serviço poderia ser efetivamente realizado. Ao invés de procurar nas páginas amarelas, o usuário poderia procurar nas páginas brancas obtendo especificamente o telefone da divisão de serviços da oficina X que é especializada na marca de seu automóvel. Analogamente seria como se o localizador do arquivo descritor tivesse sido obtido diretamente no Mecanismo de Registro, para um serviço específico de um provedor específico. Vale notar que, da mesma forma que a pesquisa nas páginas amarelas "retorna" zero ou mais telefones, a pesquisa ao Mecanismo de Registro também retornará zero ou mais localizadores e é necessário escolher qual deles é mais adequado ou conveniente para aquela situação.

Neste trabalho, conforme citação na seção 4.1.1., optou-se por pré-definir as conexões, relações e estrutura dos serviços no Repositório de Ontologias. Assim, voltando à analogia com as facilidades de uma lista telefônica, é como se o nome da empresa que faz a revisão de automóveis já estivesse devidamente identificado e anotado, por exemplo, numa lista de empresas prestadoras de serviços. Então, basta olhar a lista, localizar o nome da empresa e pesquisar a lista telefônica para obter o telefone. O Repositório de Ontologias desempenha o mesmo papel da lista de empresas prestadoras de serviços e o Mecanismo de Registro desempenha o mesmo papel da lista telefônica. Assim, o Repositório de Ontologias facilita

o trabalho de pesquisa e o Mecanismo de Registro retornará o localizador do arquivo descritor para um componente de serviço específico.

4.2 Os Repositórios da Arquitetura

Esta seção descreve os repositórios do modelo proposto.

4.2.1 Repositório de Vocabulários e Regras de Negócio

O Repositório de Regras instancia uma estrutura semelhante à *Business Vocabulary+Rules*, definida em SBVR [19]. Assim, o Repositório de Regras inclui o vocabulário de negócio e seu respectivo conjunto de regras de negócio especificado em termos daquele vocabulário de negócio. Em outras palavras, ele organiza os termos, as definições de conceitos e as regras de negócio específicos de domínios de aplicações ou que fazem sentido para uma comunidade de usuários, na realização de seus negócios. Organiza também os relacionamentos e as associações existentes entre estes elementos, através de verbos, qualificadores e questões. Para prover a habilidade na definição de conexões entre conceitos que são de interesse da organização o Repositório de Regras mantém uma estrutura semântica no nível do negócio.

A Figura 4.12 mostra o diagrama de classes para o Repositório de Regras. Pode-se ver que um vocabulário é específico de uma comunidade, para a qual o vocabulário faz sentido. Um vocabulário compõe-se de quatro conjuntos de elementos, que são os termos, verbos, qualificadores e questões. Os termos e os verbos são os elementos fundamentais do vocabulário e os qualificadores e questões são elementos complementares incluídos para facilitar a delimitação do escopo dos serviços.

Um termo representa um conceito sobre um objeto e, conforme o metamodelo SBVR, também representa uma instância (SBVR::Instantiation*)*, que por extensão é um fato (SBVR::Fact*)*, que é também uma coisa (SBVR::Thing*)*. Assim, termos servem para representar qualquer coisa, conceito, ou fato significativo para uma comunidade de

negócio. O verbo é o elemento fundamental de um predicado e, como tal, é a base para a representação de fatos.

Alguns tipos de palavras, que também incluem-se no jargão das comunidades, qualificam os termos e servem para "especializar" estes termos. Porém, os qualificadores podem ter sinônimos e isto dificulta a delimitação do escopo do serviço. A inclusão de qualificadores e respectivos sinônimos para os termos da comunidade tem o intuito de reduzir esta dificuldade. Com este mesmo objetivo, quando se trata de realização de serviço para pessoas, algumas questões pertinentes ao contexto podem criar "atalhos" na tarefa de delimitar o escopo do serviço.

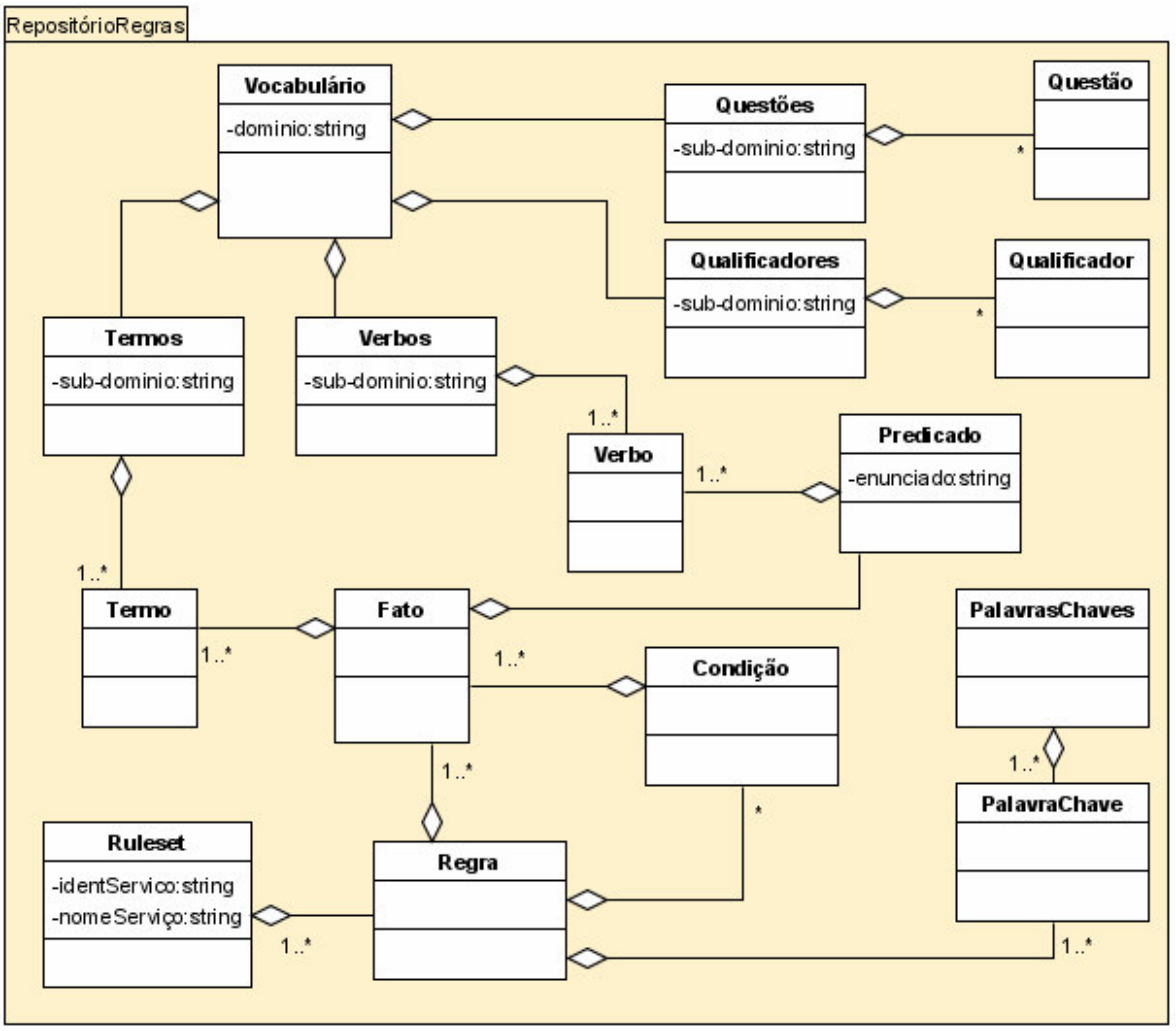

Figura 4.12 - Diagrama de Classes para o Repositório de Regras.

Estes conjuntos de elementos podem, opcionalmente, especificar um sub-domínio para representar sub-comunidades. Isto pode ser útil para algumas organizações que queiram criar sub-comunidades devido ao grande número de elementos ou às complexidades nas relações entre os atores da comunidade.

Por outro lado, um conjunto de regras é um conjunto de regras associado a um serviço específico. As regras compõem-se de palavras-chaves e fatos. As palavras-chaves são muito úteis na criação de modelos para formalizar regras livres de ambigüidades. Um fato é um período simples (ou oração absoluta) e, portanto, compõe-se de um predicado associado a um ou mais termos.

Para realçar os diferentes tipos de elementos que compõem as regras adota-se a seguinte notação: os termos serão representados em texto grifado, os fatos (verbos e predicados) em texto normal e as **palavras-chaves** em **negrito**. Assim, a regra de negócio descrita na Figura 4.13 inclui três palavras-chaves, dois termos para designar conceitos e uma designação para fato.

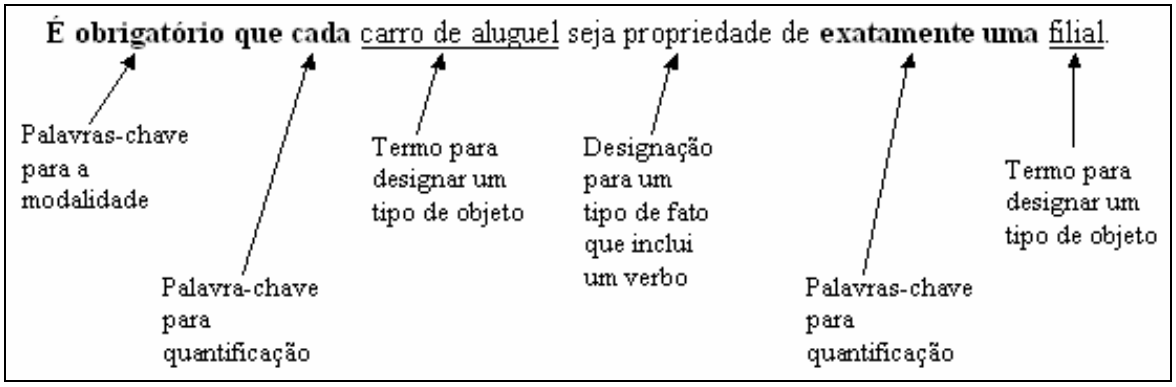

Figura 4.13 - Representação de regras no Repositório de Regras.

Todos estes elementos precisam estar devidamente organizados no repositório para que as regras de negócio sejam formalizadas referenciando estes elementos de maneira consistente. Assim, carro de aluguel e filial são termos que designam um conceito, que em última instância representam tipos de objetos.

A Figura 4.14 apresenta os casos de uso para representar as funcionalidades disponíveis no Repositório de Regras.

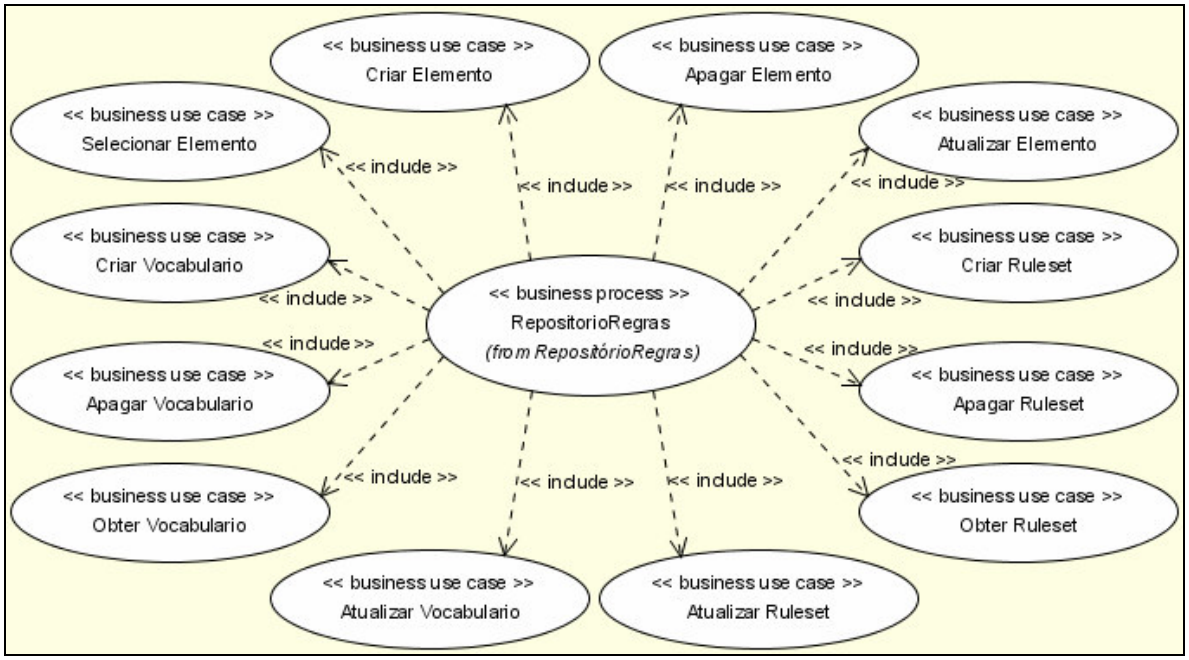

Figura 4.14 - Casos de Uso para o Repositório de Regras.

Cabe esclarecer que as quatro funcionalidades sobre elementos (Selecionar, Criar, Apagar e Atualizar) sintetizam as operações com termos, verbos, qualificadores e questões relacionados com os respectivos vocabulários e também as operações com regras, palavras-chaves, predicados, fatos, termos e verbos relacionados com os respectivos *rulesets*.

Embora os modelos de armazenamento existentes não contemplem, de maneira eficaz, as necessidades deste Repositório de Regras, existem pelo menos duas alternativas para estruturar vocabulários e regras de negócio. A primeira delas refere-se ao uso de SGBD convencional, relacional ou orientado a objetos, e a segunda ao uso de sistema gerenciador de repositório baseado no metamodelo MOF, tais como, DMOF [117], EMF [118], MDR [119] ou GRM [120]. Qualquer uma destas formas de armazenamento, apresenta vantagens e desvantagens. Os SGBDs convencionais, se por uma lado apresentam boas linguagens de definição, manipulação e consulta de dados, por outro apresentam modelos de dados que não se adequam perfeitamente a algumas das necessidades deste tipo de repositório, notadamente em relação aos padrões de intercâmbio de modelos e à possibilidade de fazer alguma inferência para auxiliar na delimitação de escopo de serviços. Já os repositórios baseados no metamodelo MOF têm como ponto forte a estruturação de vocabulários que

facilita o intercâmbio de modelos, porém não apresentam boas facilidades para acesso e consulta estruturada de vocabulários e regras de negócio.

Assim, propõe-se aprofundar as pesquisas para estruturar vocabulários e regras de negócio que culminem na especificação de um modelo de armazenamento para os Sistemas Gerenciadores de Regras de Negócio, analogamente ao modelo relacional para os SGBDs relacionais. Continuando a analogia, é necessário um modelo que suporte, por exemplo, a busca ou atualização de regras, predicados e termos em conjuntos de regras específicos, da mesma forma que um comando *select* ou *update* do modelo relacional busca ou atualiza campos em tabelas específicas. Há que se considerar também a garantia de integridade referencial entre os elementos de vocabulários e regras.

4.2.2 Repositório de Ontologias de Serviço

O Repositório de Ontologia formaliza as relações que ocorrem entre nomes e termos e seus respectivos significados associados aos componentes de serviço. Esta formalização é explícita e especifica os conceitos, as propriedades e os relacionamentos associados aos próprios componentes de serviço, cujos entendimentos são aceitos e compartilhados dentro da comunidade alvo. Ou seja, os termos usados para nomear os conceitos, as propriedades, e os relacionamentos refletem bastante o termos usados no jargão específico da comunidade ou do domínio de aplicação. Por exemplo, a ontologia de serviços para a comunidade de finanças deverá formalizar os nomes dos componentes de serviço como termos, e também as relações entre estes nomes. Portanto, além dos termos como empréstimo, taxa de juros, cliente e história de crédito, há também os nomes dos componentes de serviço que realizam ações, tais como, realizar empréstimo, verificar história de crédito e cadastrar cliente que são fundamentais para estruturar a ontologia de componentes de serviço para o domínio de finanças.

Um estrutura ontológica de componentes de serviço leva em consideração que a realização de um serviço demanda a execução de uma série de atividades que são realizadas por diferentes setores de uma empresa ou órgão público. Estas atividades, foram automatizadas em aplicações computacionais em diferentes épocas e em diferentes

plataformas e linguagens de programação. Com as disponibilidades tecnológicas atuais, o encapsulamento destas aplicações como componentes de serviço tornou-se uma alternativa com alta demanda em muitas organizações [121]. Essa tendência reforça a necessidade de se criar novos mecanismos para integrar estes componentes de serviço para contemplar os requisitos dos novos serviços. Considerando-se isto, uma ontologia de serviço comporta um conjunto de propriedades da ontologia, um conjunto de parâmetros do serviço e um conjunto de componentes de serviço. Cada componente de serviço, por sua vez, comporta um conjunto de operações com seus respectivos parâmetros. Um esquema muito simplificado desta ontologia de serviço é representado na Tabela 4.1.

Tabela 4-1 - Esquema simplificado da estrutura de um serviço.

Ontologia de Serviço Conjunto de propriedades da ontologia Conjunto de propriedades do serviço Componentes de serviço necessários Componente de serviço 1 - operação 1 lista de parâmetros ... - operação N lista de parâmetros ... Componente de serviço M - operação 1 lista de parâmetros ... - operação O lista de parâmetros

Esses relacionamentos são elementos fundamentais na ontologia de serviço porque eles conseguem representar as relações entre os termos, que neste caso são os nomes dos componentes de serviço, suas operações e seus parâmetros. A Figura 4.15 formaliza estes relacionamentos através de um diagrama de classes para o Repositório de Ontologias.

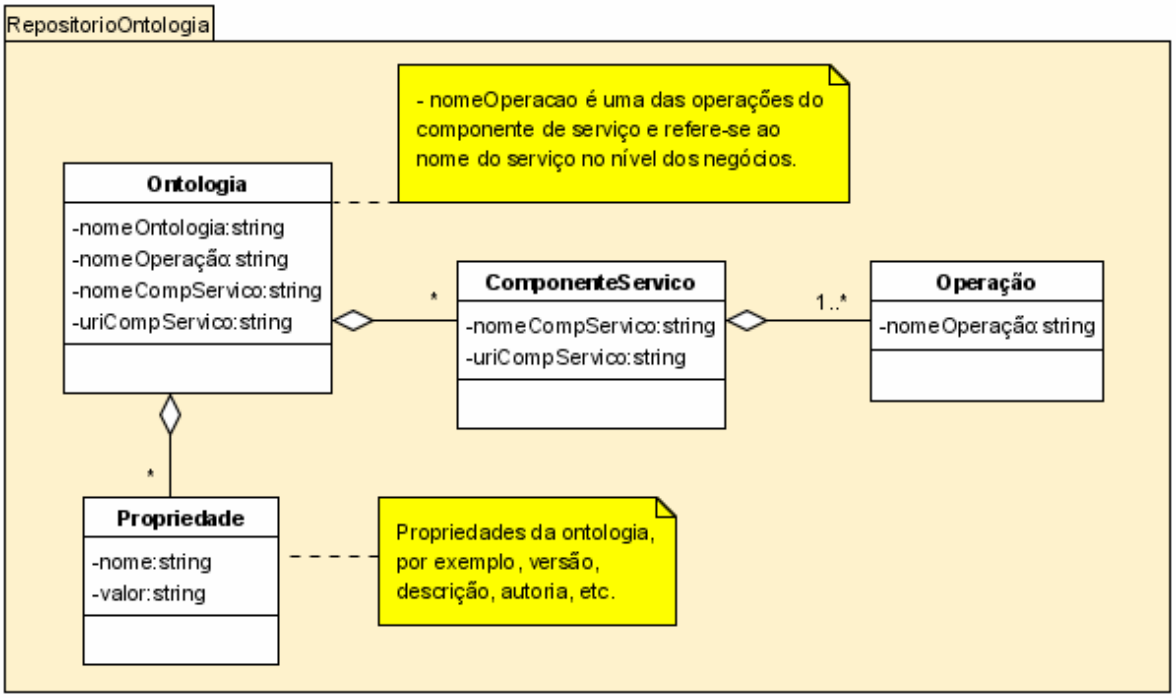

Figura 4.15 - Diagrama de Classes para o Repositório de Ontologia.

As instâncias destas representações de relacionamentos para uma ontologia de serviço específica são geradas através do IDE de Regras, em tempo de desenvolvimento ou manutenção das regras de negócio e ontologias.

A Figura 4-16 mostra o diagrama de casos de uso para o Repositório de Ontologia.

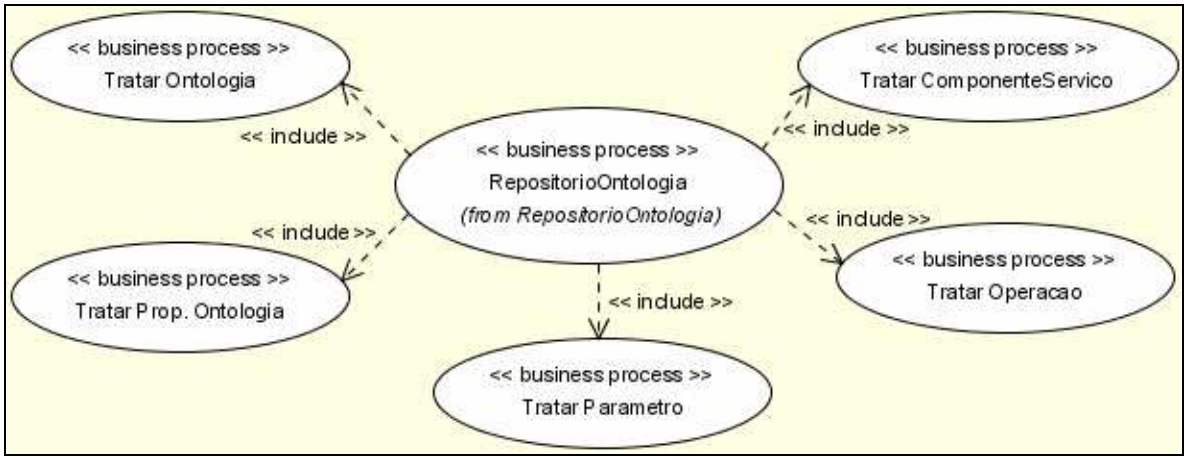

Figura 4.16 - Casos de Uso para o Repositório de Ontologia de Serviço.

Cada um destes cinco processos que compõem o Repositório de Ontologia englobam casos de uso para “criar”, “obter”, “apagar” e “atualizar” a respectiva unidade tratada pelo processo. Note que os casos de uso devem levar em consideração o domínio do objeto, sobre o qual a operação está sendo realizada, na estrutura da ontologia. Por exemplo, um parâmetro tratado pelo caso de uso ”criar” deve estar associado à operação e ao respectivo componente ou diretamente ao serviço.

Diferentemente do que ocorre no Repositório de Regras, existem várias ferramentas e padrões para estruturar ontologias dos mais variados tipos [122]. O padrão mais usado atualmente é o OWL. Embora uma ontologia possa ser armazenada em qualquer formato, o uso de um formato aderente ao metamodelo MOF facilita o intercâmbio e compartilhamento de ontologias entre diferentes sistemas. Na realidade, a utilidade de uma ontologia reside mais na sua forma instanciada que na sua forma de armazenamento, pois é na sua forma instanciada que ela pode ser manipulada por ferramentas de ontologia.

4.2.3 Repositório de Perfis de Serviço

O Repositório de Perfis armazena uma estrutura de dados chamada de Perfil de Serviço. Um Perfil de Serviço estrutura as informações necessárias para a realização de um serviço. O Repositório de Perfis é um espaço de armazenamento (*buffer*) para servir basicamente como um recurso auxiliar na implementação de um mecanismo do tipo produtor-consumidor [123] de modo a garantir assincronismo entre as operações do Preparador e do Executor. As propriedades de um Perfil de Serviço estão esquematizadas na Tabela 4.2.

Tabela 4-2 - Perfil de Serviço armazenado no Repositório de Perfis.

Propriedade	**Descrição**
Nome do serviço	Nome do serviço, no nível do negócio, a ser realizado.
Identificador do serviço	Identificador único para o serviço no contexto da comunidade.
URI do Descritor	URI do arquivo descritor para o serviço a ser realizado.
Dados Gerais	Dados e parâmetros gerais para o serviço
Componentes de serviço	Dados específicos para cada um dos componentes de serviço identificados como necessários para realizar o serviço, incluindo o link para o arquivo descritor. Um para cada componente de serviço identificado como necessário para realizar o serviço.

Deve ser lembrado que muitos dos Dados Gerais não precisam ser solicitados ao Cliente porque poderão ser obtidos automaticamente em bases de dados cadastrais das empresas ou de governos. Da mesma forma, muitos dos dados dos Componentes de Serviço serão obtidos automaticamente a partir da ontologia do serviço e dos arquivos descritores. Na maioria dos casos, os dados solicitados podem se resumir ao dado de identificação do Cliente e aos valores de alguns dos parâmetros. O diagrama de classes da Figura 4.17 mostra o relacionamento entre as classes que compõem o Repositório de Perfis de Serviço.

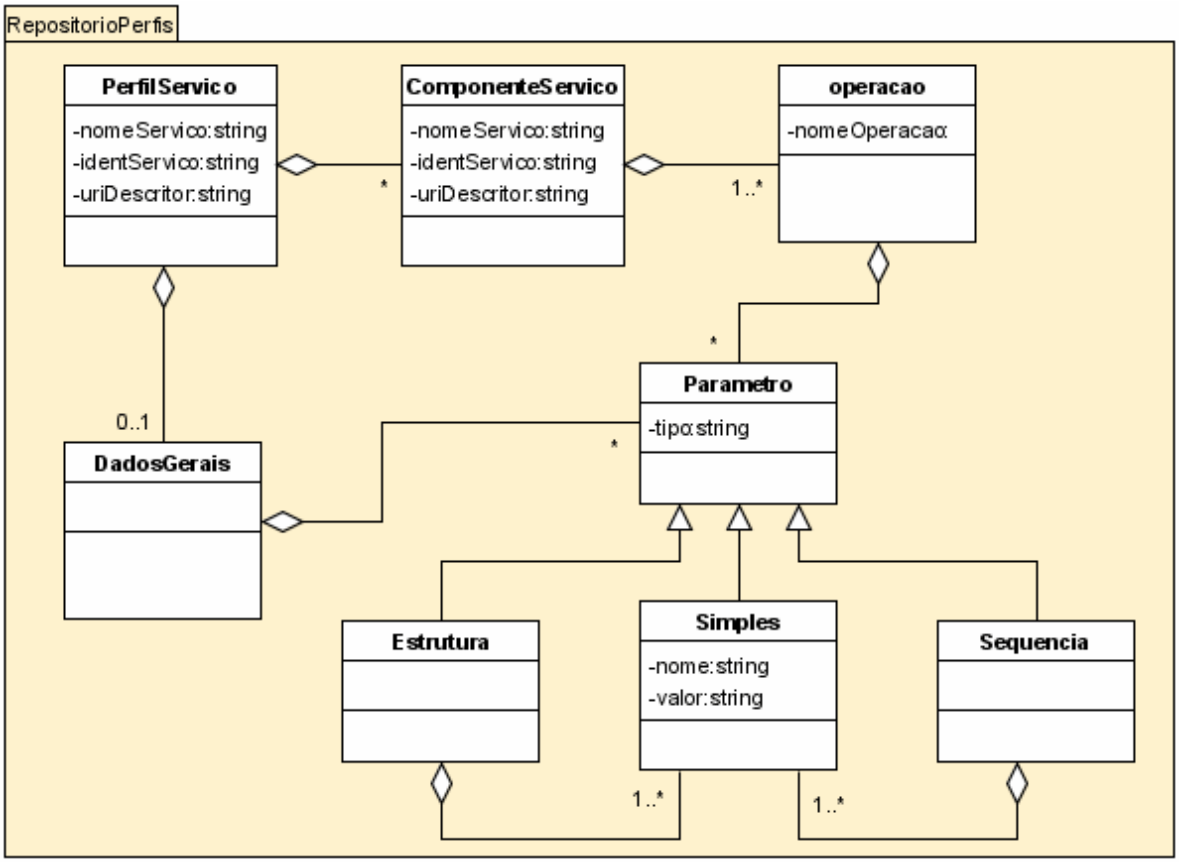

Figura 4.17 - Diagrama de Classes do Repositório de Perfis de Serviço.

O diagrama de casos de uso da Figura 4.18 mostra os processos que realizam as funcionalidades disponibilizadas pelo Repositório de Perfis.

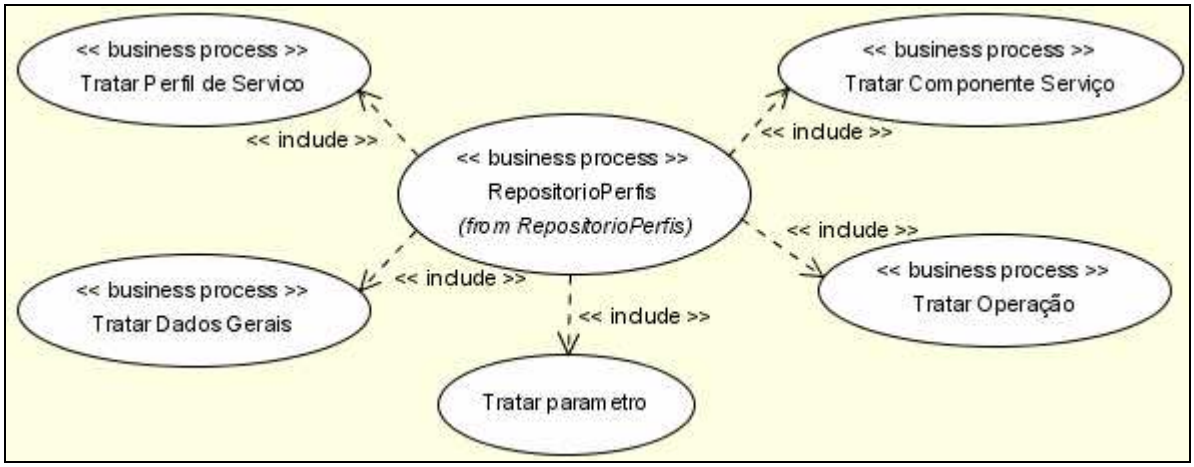

Figura 4.18 Diagrama de Casos de Uso do Repositório de Perfis.

Cada um destes cinco processos englobam casos de uso para "criar" e "obter" a respectiva unidade tratada pelo processo. Analogamente ao que ocorre no Repositório de Ontologia, aqui também os casos de uso devem levar em consideração o domínio do objeto, sobre o qual a operação está sendo realizada, na estrutura do Perfil de Serviço. Por exemplo, um parâmetro tratado pelo caso de uso "obter" deve estar associado à operação e ao respectivo componente ou diretamente ao serviço.

4.3 Transformação de modelos

Esta seção apresenta algumas propostas de transformação de modelos de regras de negócio expressas na linguagem das pessoas de negócio para modelos técnicos computacionalmente executáveis.

A especificação SBVR, no seu Anexo A, faz um posicionamento do metamodelo no contexto de MDA (*Model Driven Architecture*) [96]. Conforme pode ser visto na Figura 4.19, extraído de SBVR [19] os modelos de negócio, aderentes ao metamodelo SBVR estão no nível CIM (*Computation Independent Model*) de MDA. Portanto, estes modelos, que

incluem as regras de negócio, vocabulários, conceitos e fatos, descrevem os negócios e não os sistemas de informação que os suportam.

Em MDA, sistemas de informação são especificados usando modelos independentes de plataforma (PIM – *Platform Independent Model*) e modelos específicos de plataforma (PSM – *Platform Specific Model*). Embora esteja fora do escopo do metamodelo SBVR, a própria especificação dá indicações sobre a necessidade de se criar regras de transformação de modelos de negócio, aderentes a SBVR, para modelos PIM. Feito isto, o ciclo deve ser completado com transformações de modelos PIM para modelos PSM.

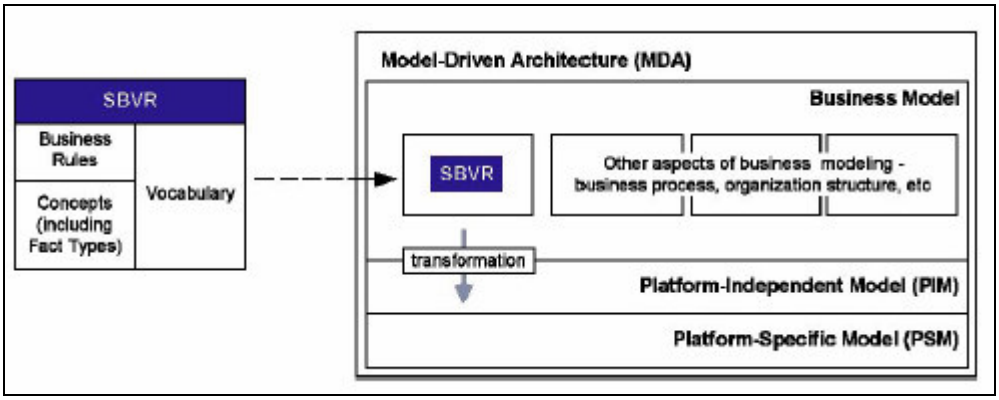

Figura 4.19 - Posicionamento de SBVR em MDA [19].

As sub-seções a seguir discutem regras de transformação de modelos de regras de negócio. Vale lembrar que, estas regras de transformação de modelos são regras técnicas no contexto da Tecnologia da Informação e, portanto, no mínimo, estão num patamar de abstração diferente do das regras de negócio, que são os objetos desta tese. Por outro lado, se considerarmos que os artefatos usados para tratar estas regras de transformação são de domínio da comunidade de TI em uma sub-área da Engenharia de Software, então as regras de transformação são também regras de negócio para uma sub-comunidade da Engenharia de Software.

Embora a especificação do metamodelo SBVR dê algumas indicações sobre a transformação de modelos aderentes a SBVR para outros modelos, não há ainda propostas concretas com este objetivo. A especificação propõe a criação de regras de transformação de modelos SBVR para modelos de regras executáveis de qualquer máquina de regras.

Conforme sugere a especificação SBVR, às vezes é necessário estender o metamodelo SBVR com a inclusão de novos predicados, por exemplo, para poder tratar semânticas contempladas no metamodelo SBVR, mas não contempladas no modelo alvo.

4.3.1 Regras de Transformação de modelo SBVR para modelo SWRL

Embora ainda não haja implementações concretas de máquina de regras para SWRL, mesmo porque a linguagem ainda está em fase de especificação, apresenta-se a seguir um exercício de como seriam algumas das regras de transformação de modelos SBVR para modelos SWRL. Vale lembrar que SWRL é baseada em RuleML e OWL e, portanto, um código em SWRL contém declarações RuleML e OWL.

1. **Um elemento** SBVR::Instantiation **deve** ser transformado **para** owl:Class

   Deve-se contemplar inclusive as eventuais restrições sobre propriedades.

2. **Um elemento** SBVR::Fact **deve** ser transformado **para** owl:Class e owl:ObjectProperty.

   A aridade do SBVR::Fact determina o número de owl:Class, ou seja se for um *fact-type* unário terá somente um owl:Class, se for um *fact-type* binário terão duas owl:Class, e assim por diante.

3. **Um elemento** SBVR::Text **deve** ser transformado **para** owl:DatatypeProperty
4. **Um elemento** SBVR::Integer **deve** ser transformado **para** owl:DatatypeProperty
5. **Uma** regra **com** proposição de obrigatoriedade (regra com *deontic operator*) **deve** ser transformado **para** <ruleml:imp> **com** indicação para a obrigatoriedade.

   Ou seja, os fatos (SBVR::Fact) que compõem as pré-condições da regra com cláusulas de obrigatoriedade devem ser incluídos como variáveis (<ruleml:var>), classes (<owl:Class>), propriedades de objetos (<swrl:individualPropertyAtom>), e propriedades de dados (<swrl:datavaluedPropertyAtom>) no **corpo da regra** (ruleml:_body>). De maneira semelhante, os fatos (SBVR::Fact) que compõem as pós-condições da regra com cláusulas de obrigatoriedade devem ser incluídos como variáveis (<ruleml:var>), classes (<owl:Class>), propriedades de objetos

(<swrl:individualPropertyAtom>), e propriedades de dados (<swrl:datavaluedPropertyAtom>) na **cabeça da regra** (ruleml:_head>).

6. **Uma** regra **com** proposição de proibição (deontic operator) **deve** ser transformado **para** <ruleml:imp> **com** indicação para a proibição.
7. **Uma** regra **com** proposição de necessidade (alethic operator) **deve** ser transformado **para** owl:ObjectProperty **com** indicação para a necessidade.
8. **Uma** proposição com cláusula de possibilidade (alethic operator) **deve** ser transformado **para** owl:ObjectProperty **com** indicação para a possibilidade.

Uma vez transformado todos os elementos de um modelo é necessário eliminar redundâncias geradas durante as transformações. Estas redundâncias ocorrem porque duas ou mais regras que tratam o mesmo termo podem, cada qual, gerar a sua classe para o mesmo termo. Neste caso, é necessário realizar otimizações entre as classes e especializações de classes OWL de modo a eliminar redundâncias. Cabe esclarecer que este conjunto de regras não está completo, por se tratar de uma proposta. Porém, o conjunto exercitado mostra que a transformação completa é perfeitamente realizável, principalmente se consideradas as facilidades de extensão do metamodelo SBVR.

4.3.2 Regras de Transformação de modelos SBVR para modelos Jess

Apresenta-se a seguir um exercício de transformação de modelos SBVR para modelos da máquina de regras Jess.

1. **Um elemento** SBVR::Instantiation **deve** ser transformado **para** uma (definstance) **do respectivo** (defclass).

   Ou seja, **a cada** termo **deve** ser criada **uma** instância **da** classe associada **ao** termo.

2. **Um elemento** SBVR::Fact **deve** ser transformado **para uma** (definstance) **do respectivo** (defclass). **E cada** SBVR::Instatiation **associado a** SBVR::Fact **deve** ser transformado **para as** (definstance) **dos respectivos** (defclass).

   Ou seja, **a cada** fato **deve** ser criada **uma** instância **da** classe associada **ao** fato **e para cada** termo **do** fato **deve** ser criada **uma** instância **da** classe associada **ao**

termo. A aridade do SBVR::Fact determina o número de (definstance), ou seja se for um *fact-type* unário terá somente uma instância, se for um *fact-type* binário terão duas instâncias, e assim por diante.

3. **Um elemento** SBVR::Text **deve** ser transformado **para uma** (definstance) do respectivo (defclass).
4. **Um elemento** SBVR::Integer **deve** ser transformado **para** (definstance) **do respectivo** (defclass)
5. **A cada** atributo **de** instância **deve** ser atribuído **um** valor *default*.
6. **Cada** valor obtido **deve** ser atribuído **ao respectivo** atributo. (Atribuir aos atributos os valores possíveis já obtidos. Estes valores foram obtidos a partir da interação com o usuário ou automaticamente de bases cadastrais.)
7. **Cada** SBVR::Rule simples **deve** ser transformado **para** defrule **com os** atributos dos objetos **referenciados nas** regras.
8. **Cada** SBVR::Rule com obrigatoriedade (*deontic operator*) **deve** ser transformado **para** defrule **com** mecanismo de encadeamento regressivo **para** satisfazer pré-condições.

   Assim, se regra contém clausula deôntica (“É obrigatório que“ ou “... deve ...“) então

   - Pré-condição que não estiver associada a razão deve ser transformada para o padrão (***pattern)*** **Jess** equivalente à pré-condição, com variáveis para todos os atributos relevantes do respectivo objeto.
   - Incluir declaração de encadeamento regressivo para este **padrão Jess.**
   - Incluir a regra respectiva (com prefixo “find-“) para realizar o processo encadeamento regressivo. Esta nova regra deverá ter no LHS o mesmo padrão anterior com prefixo “need-“ e variáveis “não casadas“ (ou não associáveis a valores fornecidos) em aberto, ou seja, com “?“. No RHS deverá ter chamada de um serviço que checará o estado de um objeto e caso não satisfeito realizará uma ação para mudar o estado do objeto.).

O caráter deôntico (obrigação ou proibição) de uma regra SBVR é garantido pela representação de pré-condições como fatos no LHS de uma regra em Jess.

Se algum destes fatos não for verdade outra regra é disparada, usando-se o mecanismo encadeamento regressivo de Jess para satisfazer a precondição associada ao fato.

Novamente, após a transformação de todos os elementos de um modelo é necessário eliminar redundâncias geradas durante as transformações. As redundâncias ocorrem quando duas ou mais regras ao tratar do mesmo termo, cada qual, gera a sua classe para o mesmo termo. Assim, é necessário realizar otimizações entre as classes e especializações de classes Jess de modo a eliminar redundâncias. Aqui também cabe esclarecer que este conjunto de regras não está completo, por se tratar de uma proposta. Porém, o conjunto exercitado mostra que a transformação completa é perfeitamente realizável, principalmente se consideradas as facilidades de extensão do metamodelo SBVR.

4.4 Considerações Finais do Capítulo 4

Este capítulo apresentou conceitos para desenvolvimento e execução de serviços, baseados em regras de negócio. Esses conceitos fornecem idéias para solucionar problemas relacionados com a pouca flexibilidade na manutenção e na construção de sistemas computacionais inseridos no contexto das mudanças rápidas e dinâmicas que ocorrem no mundo dos negócios na atualidade.

A plataforma de desenvolvimento de aplicações inclui um IDE com um editor para definir e gerenciar regras de negócio e vocabulários na linguagem das pessoas de uma comunidade de negócio. Os elementos que compõem um vocabulário são os termos, verbos, qualificadores e questões que, juntamente com as regras de negócio, formam a base para a definição do Repositório de Regras. O IDE mantém um conjunto de modelos para facilitar a definição de regras que, resgatando elementos do Repositório de Regras e descobrindo componentes de serviços num Mecanismo de Registro, define uma ontologia de serviço à qual a regra em definição está relacionada. Esta ontologia de serviço é armazenada no Repositório de Ontologias.

A plataforma de execução de serviços inclui um módulo (Preparador) para preparar a execução e outro (Executor) para executar o serviço conforme configurado pelo Preparador.

A preparação da execução consiste em delimitar o escopo do serviço através de interações com o Cliente e com os repositórios de regras e ontologias, verificar a disponibilidade de componentes de serviço para executar o serviço e consistir os parâmetros de arquivos descritores. A execução do serviço consiste em instanciar uma máquina de regras, traduzir regras descritas na linguagem de pessoas de negócio para formatos executáveis, gerar programas clientes para os componentes de serviço e executar as regras considerando os recursos disponibilizados.

Portanto, a plataforma de desenvolvimento provê facilidades para criar e manter regras de negócio de maneira flexível, de modo que as mudanças que ocorrem nos negócios refletem-se rapidamente nos sistemas computacionais. Por outro lado, a plataforma de execução provê facilidades para realizar serviços, a partir da descoberta e execução de componentes de serviço, sob controle de regras de negócio expressas na terminologia das pessoas de negócio.

Capítulo 5

A Prova de Conceito

Este capítulo fornece uma visão geral de como foi realizada a prova de conceito para a abordagem proposta neste trabalho. Assim, na seção 5.1 apresentam-se as condições de contorno e as decisões sobre como os aspectos da proposta são considerados na prova de conceito. Na seção 5.2 apresenta-se o cenário usado para exercitar a prova de conceito, incluindo o uso do IDE na criação e manutenção de regras de negócio. Nas seções 5.3 descrevem-se alguns aspectos da criação e edição dos elementos tratados pelo Repositório de Regras e Repositório de Ontologia. Nas seção 5.4 descrevem-se alguns aspectos das etapas de preparação do serviço e de execução do serviço.

5.1 Condições de Contorno

O exercício de implementação de ferramentas para suportar a proposta de desenvolvimento de regras e execução de serviços abordada nesta tese exige um grande esforço no detalhamento de projeto dos módulos e na programação. Assim, a prova de conceito para esta proposta, que inclui (i) um IDE de regras de negócio e ontologias de serviços combinado com (ii) um ambiente de execução de regras de negócio, foi obtida implementando-se alguns aspectos da proposta e simulando vários outros para simplificar e reduzir o esforço de implementação.

As condições de contorno para definir os aspectos considerados essenciais para esta prova de conceito foram determinadas à luz do volume de esforço necessário e pelo grau de complexidade ou exeqüibilidade das atividades envolvidas. Assim, considerando estas condições de contorno, as atividades foram agrupadas da seguinte forma:

1. Atividades consideradas de alta exeqüibilidade não foram implementadas, definindo-se apenas suas interfaces e exercitando-se com conjuntos de dados pré-definidos.
2. Algumas das atividades também consideradas exeqüíveis, mas que exigiriam muito esforço de programação também não foram implementadas, simulando-se seus resultados através de arquivos em formato texto ou XML.
3. Para outras atividades que ainda apresentam questões a ser analisadas e discutidas foram somente fornecidas algumas sugestões de como poderiam ser implementadas.
4. As atividades que tratam dos aspectos considerados essenciais nesta prova de conceito são aquelas que são necessárias para exercitar o ciclo completo de execução de um serviço com as respectivas chamadas às regras de negócio e as respectivas invocações de Serviços Web.

Considerando-se os módulos e as atividades no diagrama de seqüência da Figura 4-2 pode-se mapeá-los nos grupos de atividades da seguinte forma:

1. Atividades de alta exeqüibilidade - Assumiu-se que as atividades relacionadas com a implementação do Cliente, da estrutura de navegação do Cliente na forma como foi idealizada neste trabalho e de publicação de Serviços Web no Mecanismo para Registro de Serviços Web são consideradas de alta exeqüibilidade.
2. Atividades com muito esforço de programação – Neste grupo encontram-se todas as atividades que compreendem a criação e edição de regras de negócio usando o IDE de Regras e a preparação do serviço. Portanto, embora o IDE de Regras esteja especificado e exemplificado o seu uso através de protótipos de interface ele não foi implementado. Da mesma forma, embora as ações do Preparador estejam especificadas e exemplificada a interação com o Cliente e com os repositórios para delimitar o escopo do serviço, o Preparador também não foi implementado.
3. Atividades com questões em aberto – Neste grupo encontram-se as atividades que englobam as operações e estruturação dos três repositórios. As informações armazenadas nestes repositórios merecem uma abordagem não convencional na estrutura e forma de manipulação destas informações. Assim, optou-se por discutir e

analisar algumas possibilidades e sugerir algumas alternativas de implementação, tais como, o uso de SGBDs não convencionais, repositórios baseado no metamodelo MOF e repositórios de ontologia, conforme podem ser vistos na seção 4.2.

4. Atividades essenciais – Neste grupo estão algumas das atividades do módulo Executor, que foi implementado em Java. As interações com outros módulos não implementados foram realizadas na forma de simulações ou apenas sugeridas e assumindo-se dados e resultados pré-definidos.

Portanto, foram implementados somente alguns aspectos do módulo Executor, aspectos estes que realizam atividades essenciais para fins da prova de conceito. As principais atividades realizadas pelo Executor podem ser resumidas na seguinte lista:

- Busca Perfil do serviço
- Busca regras (rule set)
- Gera *proxy* WS baseado no arquivo WSDL (Web Services Description Language) e compila-o
- Traduz regras (SBVR-Jess)
- Instancia máquina Jess e carrega regras em memória
- Gera fatos baseados nos eventos de vida
- Executa máquina Jess

Desta lista, assumiu-se como realizadas as quatro primeiras atividades, considerando-se como disponíveis para oExecutor os seus resultados. Assim, assumiu-se que:

- o serviço já fora devidamente identificado (Solicitação de Segunda Via de RG) pelo Preparador e o respectivo Perfil de Serviço já fora recuperado do Repositório de Perfis.
- Baseado no Perfil de Serviço recuperou-se do Repositório de Regras o conjunto de regras associado ao serviço de Solicitação de Segunda Via de RG.
- Baseado no arquivo descritor do Serviço Web RG e nos dados do Cliente contidos no Perfil de Serviço, gerou-se o programa-cliente para o Serviço Web. O programa-cliente foi compilado.

- Usando um algoritmo de transformação de regras, traduziu-se as regras, aderentes ao metamodelo SBVR para o modelo da máquina de regras Jess.

5.2 Cenário de Uso

O cenário escolhido insere-se no âmbito da comunidade de Identificação Civil do Estado de São Paulo, e o serviço escolhido o de "solicitar 2a via da carteira de identidade, por perda da original". Embora seja um serviço já oferecido à população, ele não é realizado da forma integrada e automatizada conforme sugere-se neste trabalho. O serviço foi escolhido por apresentar uma grande variabilidade de dados de contexto e, assim, ser bastante adequado aos propósitos desta tese. A presença desta grande variabilidade de dados de contexto deve-se ao fato de que os sistemas legados que manipulam estes dados foram construídos por diferentes setores do governo, em diferentes épocas e usando diferentes plataformas e linguagens de desenvolvimento [124]. Por todos estes aspectos, estes sistemas legados disponibilizam aplicações já bastante estabilizadas no que se refere a mudanças e, portanto, ideais para serem encapsuladas como serviços Web.

A proposta e os conceitos para desenvolvimento e execução de serviços baseados em regras de negócio podem ser aplicados em qualquer área de negócio, notadamente naquelas categorizadas como B2B, B2G, G2C e G2G. O cenário aqui usado inclui-se numa categoria mista com aplicações fortemente G2C, mas com ramificações em B2G e G2G. É fácil notar o aspecto G2C neste cenário, pois o serviço de "solicitação de segunda via de RG" é um serviço oferecido pelo governo ao cidadão. O aspecto B2G pode ser percebido quando envolve alguma transação entre empresa e governo. No serviço aqui escolhido o cidadão deve pagar uma taxa de serviço para a emissão da segunda via do RG em algum banco comercial, em favor do governo. O aspecto G2G pode ser percebido quando envolve operações entre diferentes setores ou esferas de governo, que é o caso, neste serviço.

Antes da descrição do cenário de uso, propriamente dito, inclui-se um exercício de uso do editor, que é o principal componente do IDE, para facilitar o entendimento de como as regras de negócio são definidas.

5.2.1 O Uso do IDE

Conforme visto no capítulo 4, o IDE comporta um editor de regras de negócio, vocabulários e ontologias de serviço. A edição de regras baseia-se no uso de modelos associados aos diferentes tipos de regras de negócio definidos. Assim, para facilitar o desenvolvimento de regras de negócio, esses modelos são incorporados num editor gráfico, conforme mostra o protótipo de interface da Figura 5.1.

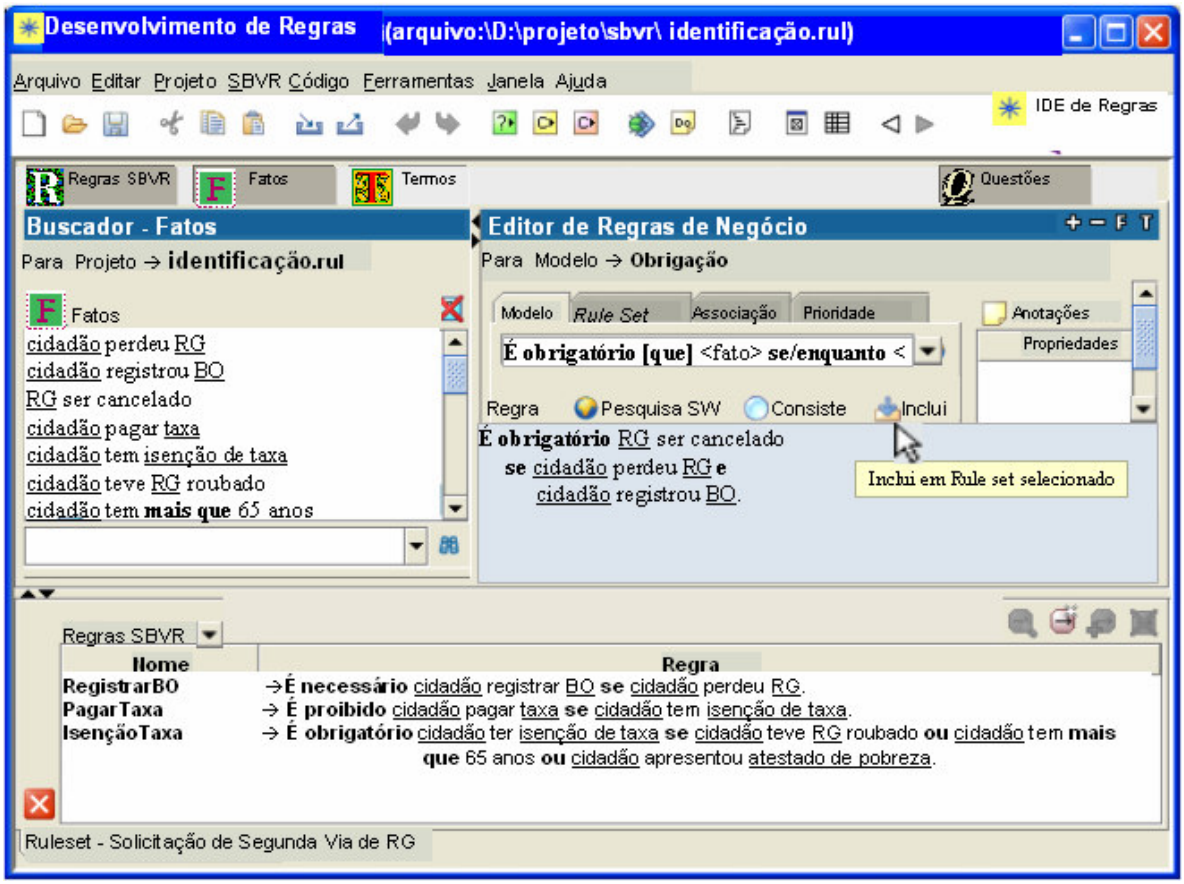

Figura 5.1 - Protótipo de Interface para o Editor de Regras.

Assim, para "Definir Regra"

- O analista de negócio seleciona no Editor o vocabulário da sua comunidade (Arquivo→Abrir – identificação.rul) e o conjunto de regras do qual a regra fará parte (Editor de Regras de Negócio→ficha *Rule Set*, seleciona o conjunto de regras e as eventuais regras já definidas aparecem no painel inferior).

- O analista de negócio seleciona o modelo desejado.
- O Editor busca o modelo e apresenta-o numa janela do Editor.
- Quando o analista de negócio aponta para um campo do modelo o Editor busca a lista associada ao campo. Por exemplo, no modelo **"É obrigatório** [**que**] <fato> [**se/enquanto** <condição>]", apontando-se para <fato> ou <condição> busca-se a lista de predicados relativos ao vocabulário selecionado. Uma vez escolhido o predicado e apontando-se para um de seus campos busca-se no vocabulário a lista de termos que fazem sentido naquele contexto.
- Alguns <fatos> e <condições>, para se tornarem verdadeiros, pressupõem a realização de alguma ação, que pode ser realizada por um serviço Web. O analista de negócio seleciona "Pesquisar SW" e, com as palavras contidas num <fato> ou <condição> o Editor executa "Pesquisar URI de Descritor" para descobrir se há algum serviço Web que realiza a ação.
- Para cada URL de arquivo WSDL válido o Editor busca o respectivo arquivo WSDL.
- O Editor abre cada arquivo WSDL e obtém os elementos-chave, tais como, o URL do serviço Web e os nomes das operações. Estes elementos são inseridos na Ontologia de Serviço ao qual a regra em definição deve ser associada.
- O analista de negócio pode selecionar "Consiste" para verificar a consistência e a sintaxe.
- Por fim, o analista de negócio seleciona "Incluir" no conjunto de regras que está sendo tratado.

Exemplos típicos de termos de negócio (*business terms*) de um vocabulário relacionado com a terminologia da comunidade de identificação civil podem ser consolidados nas informações contidas na Tabela 5-1.

Tabela 5.1 - Exemplos de termos associados com Perda de RG.

Termo de negócio	Descrição ou necessidade
Carteira de Identidade	Carteira de identificação civil com validade nacional
RG	Sigla para Registro Geral. Semanticamente idêntico à Carteira de Identidade
Identidade	Semanticamente idêntica à Carteira de Identidade
Segunda Via de RG	Qualquer via extra da Carteira de Identidade, excetuando-se a primeira.

	Qualquer cidadão pode solicitar a emissão da Segunda Via de RG a qualquer instante e, praticamente, por qualquer razão.
Novo RG	Semanticamente idêntico à Segunda Via de RG
Boletim de Ocorrência	Documento oficial para relatar uma ocorrência, por exemplo, a perda de um RG.
BO	Semanticamente idêntico ao Boletim de Ocorrência.
Ficha Criminal	Documento oficial para registro de ocorrências criminais de um cidadão armazenado no sistema de Identificação Criminal
Impressão digital	Documento, eletrônico ou não, para armazenar impressões digitais do cidadão.

A Tabela 5.2 mostra algumas decisões de negócio e as respectivas regras de negócio associadas à solicitação de Segunda Via de RG, por perda. Este conjunto de regras de negócio associadas a um determinado serviço definem o que em SBVR denomina-se conjunto de regras (*rule set)*. E, assim, uma necessidade do cidadão ficará associada a um conjunto de regras.

Tabela 5.2 – Conjunto de regras associado à solicitação de Segunda Via de RG, por perda.

Questões/Decisões de Negócio	**Regras de Negócio para pessoas de negócio**	**Possível Formalização (*rule set*)**
O cidadão registrou um BO? ou BO está registrado?	Todo cidadão cujo RG foi perdido precisa registrar um BO.	**É necessário** cidadão registrar BO **se** cidadão perdeu o RG.
RG está cadastrado?	Se dados do RG existem então ele está cadastrado.	**Se** dados RG existem então RG está cadastrado.
RG está cancelado? (pós-condição)	Um RG deve ser cancelado se o mesmo foi perdido, BO está registrado e RG está cadastrado.	**É obrigatório** RG ser cancelado **se todas as seguintes situações ocorrerem:** • cidadão perdeu o RG • cidadão registrou BO • RG está cadastrado
O cidadão solicitou a Segunda via do RG? Ou Segunda via do RG foi solicitada?	A segunda via do RG pode ser solicitada nas seguintes situações: • RG está antigo ou danificado • cidadã casou-se ou descasou-se e quer mudar o nome • cidadão quer mudar a foto • cidadão quer mudar a assinatura • cidadão perdeu o RG • cidadão teve seu RG roubado • cidadão teve seu RG furtado • RG está cancelado	**É possível** Segunda Via do RG ser solicitada **se pelo menos umas das seguintes situações ocorrerem:** • RG está antigo ou danificado • cidadão quer mudar nome casou-se ou descasou-se e • cidadão quer mudar a foto • cidadão quer mudar a assinatura • cidadão perdeu RG • cidadão teve seu RG roubado • cidadão teve seu RG furtado • RG está cancelado
O cidadão deve pagar a taxa de serviço para que a Segunda Via de RG seja emitida.	Todo cidadão deve pagar a taxa de serviço para que a Segunda Via de RG seja emitida se ele perdeu o RG e não está isento do pagamento da taxa de serviço.	**É obrigatório** cidadão pagar a taxa para emissão de Segunda Via de RG **se todas as seguintes situações ocorrerem**: • cidadão perdeu o RG • cidadão não tem isenção de taxa para

		emissão de Segunda Via de RG.
	Um cidadão que teve seu RG roubado, ou tem mais que 60 anos, ou apresentar Atestado de Pobreza, ou está desempregado há mais de 3 meses, está isento de pagamento taxa de serviço para emissão de Segunda Via de RG	**É obrigatório** cidadão ter isenção de taxa para emissão de Segunda Via de RG **se** cidadão teve RG roubado **ou** cidadão tem mais de 65 anos **ou** cidadão apresentou atestado de pobreza **ou** cidadão está desempregado há mais de 3 meses.

5.2.2 Portal Web como Cliente

O cenário escolhido inclui um cidadão que está diante de um portal Web de governo. Esta é a situação em que o módulo Cliente da arquitetura é instanciado como um portal Web e o cidadão interage com o portal Web para satisfazer sua necessidade.

Considere o portal Web que, baseado em eventos de vida, começa a tratar as necessidades do cidadão com orientações do seguinte tipo:

Orientação 1 - "Por favor, entre com uma frase curta indicando **o que deseja** ou **o que lhe aconteceu:** ....."

Diante de uma orientação deste tipo o cidadão poderia descrever **uma necessidade ou um desejo**, tais como, **"**quero pagar uma conta", "quero obter certidão negativa", "preciso comprar uma passagem", "desejo tirar carteira de identidade", "necessito renovar licença de veículo", ou **um evento de vida, um incidente ou insatisfação**, tais como, "minha conta de luz está muito alta", "meu carro foi roubado", "perdi minha carteira de identidade", "meu filho nasceu", etc. Ao invés desta abordagem de interação textual em linguagem natural, uma alternativa também realista e moderna seria a inclusão de mecanismos de processamento de fala para suportar o diálogo no tratamento das necessidades do cidadão. Esforços neste sentido são despendidos pro grupos de pesquisa da Carnegie Mellon University com mecanismos de reconhecimento de fala denominado Sphinx [125] e AIT [126], reforçados pelos exemplos de diálogos mostrados em RoomLine [127], e conforme atestam a sua viabilidade as ferramentas denominadas Dragon NaturallySpeaking 9 da Nuance [128], e ViaVoice da IBM [129].

Conforme visto no capítulo 4, o portal Web é controlado pelo Preparador, que tem como função principal preparar o serviço para realizar uma necessidade. Portanto, o Preparador deve fazer uso da facilidade do diálogo orientado por regras de negócio na tarefa de descobrir a necessidade.

5.2.3 A Obtenção do Vocabulário

Assim, supondo que o cidadão entre com "Eu perdi meu RG" para a orientação anterior (Orientação 1), o Preparador consulta o Repositório de Regras e obtém o vocabulário relacionado com a perda da carteira de identidade para continuar o processo de descoberta do serviço. Este vocabulário contém os termos, verbos, qualificadores e questões associados à "perda de RG", tais como os mostrados na Tabela 5.3. Note que, o importante é a captura dos verbos (em qualquer conjugação) e dos termos incluídos nas entradas do cidadão. Se os verbos e termos contidos na frase de entrada do cidadão coincidirem com os do vocabulário, então a razão para um eventual serviço já estará sendo definida. Assim, em "Eu perdi meu RG" o verbo é "perder", os termos são "cidadão" (Eu) e "RG" e a razão preliminar é "cidadão perdeu RG".

Tabela 5.3 - Elementos do vocabulário relacionado com a "perda de RG".

Verbos relacionados com a Necessidade, Incidente ou Insatisfação	**Termos e Qualificadores**	**Perguntas e orientações subseqüentes para definir o escopo do serviço**	**Razão ou situação ou termos relacionados com a razão ou situação**
Perder, Perdi, Foi perdido, Foi furtado	Carteira de identidade, Identidade, RG, cidadão, BO, Boletim de Ocorrência, cancelamento de RG, solicitação de segunda via de RG, cadastramento de RG, etc.	"Por favor, entre com os dados de seu RG (nome, numero, estado emissor, e se possível a data de emissão):....." "Deseja solicitar a segunda via de RG?" -Sim (necessidade) -Não (4) 4- Já registrou um BO? -Sim -Não (5) 5- Você precisa urgentemente registrar um BO! Deseja registrar um BO neste momento? -Sim (necessidade) -Não (6) 6- Seu RG deve ser cancelado para evitar uso indevido. 7- Seu RG será cadastrado com os dados do BO.	1- Cidadão perdeu RG, ou RG do cidadão foi furtado.

A partir deste vocabulário, o Preparador, que suporta um mecanismo para tratar diálogos orientados por regras de negócio, provê as seguintes novas orientações:

Orientação 2 - "Por favor, entre com os dados de seu RG Nome: ..... Numero do RG:..... Estado emissor: E, se possível a data de emissão (dd/mm/aaaa):....."

Orientação 3 - "Deseja solicitar a segunda via do RG? (S/N): ....."

5.2.4 A Descoberta do Serviço

Assumindo-se que o cidadão entre com "José Silva" como nome, "123" como número do RG e "São Paulo" como estado emissor para a Orientação 2 e responda "sim" na Orientação 3, o Preparador terá elementos para concluir que o desejo do cidadão "José Silva" é a realização do serviço "solicitar a segunda via do RG que foi emitido no estado de São Paulo" e a razão para solicitar o serviço é a "perda do RG". Neste contexto, a questão "Deseja solicitar a segunda via do RG?*"* é bastante providencial porque a realização deste desejo pode incluir a realização de vários outros desejos do cidadão, que de qualquer forma ele gostaria de realizar, tais como, registrar um BO, cancelar a Carteira de Identidade, atualizar dados da Carteira de Identidade, trocar a foto da Carteira de Identidade, mudar a assinatura da Carteira de Identidade, etc.

Numa outra situação, caso o cidadão mantenha as respostas à Orientação 2 e responda "não" à Orientação 3 o Delimitador dará a seguinte orientação:

Orientação 4 – "Já registrou um BO?"

E, assim, cada serviço teria uma estrutura de navegação nas orientações ou estrutura de diálogos orientada por regras de negócio para conduzir a descoberta da necessidade do cidadão.

Supondo que na "Orientação 1", ao invés de "*Eu perdi meu RG.*" o cidadão tenha entrado com "***Eu quero tirar a segunda via de meu RG***" o vocabulário recuperado do Repositório de Regras conteria os elementos mostrados na Tabela 5.4. Assim, uma

pergunta que deveria ser feita, num dado instante, para delimitar o escopo do serviço seria algo como "***Por que deseja solicitar a segunda via de seu RG?***". Muito provavelmente a resposta do cidadão seria "***Porque eu perdi meu RG.***".

Tabela 5.4 - Elementos do vocabulário para "solicitação de Segunda Via de RG".

Verbos relacionados com a Necessidade, Incidente ou Insatisfação	**Termos e Qualificadores**	**Perguntas e orientações subseqüentes para definir o escopo do serviço**	**Razão ou situação ou termos relacionados com a razão ou situação**
Quero, quero obter, quero tirar, desejo, preciso, necessito	carteira de identidade, identidade, RG, nova, outra, segunda via	Entre com o número do RG, Nome, Estado Emissor, :..... Por que deseja solicitar a segunda via da carteira de identidade? Respostas: -Porque eu perdi minha carteira de identidade (1). - Porque a minha carteira de identidade foi roubada (2). - Porque a minha carteira de identidade está velha (3).	1- Cidadão perdeu RG, ou RG do cidadão foi furtado. 2- RG do cidadão foi roubado. 3- RG do cidadão está velho, ou rasgado, ou danificado.

Portanto, qualquer que tenha sido a forma de navegação o que se obteve foi a identificação do serviço para realizar a necessidade do cidadão e a razão que embasa essa necessidade. Todas estas informações capturadas do Cliente e outras que são obtidas automaticamente de bases de dados cadastrais são, a todo instante, mapeadas na estrutura do Perfil de Serviço.

5.2.5 A Verificação da Disponibilidade dos Serviços Web

Uma vez identificado o serviço, para verificar a disponibilidade de eventuais serviços Web, o Preparador pesquisa o Repositório de Ontologia e obtém a ontologia que refere-se ao serviço. Basicamente a ontologia conterá informações que identificam a ontologia, o Serviço Web que inclui a operação que realiza o serviço, a operação em si, a URL para arquivo WSDL do serviço e os nomes do Serviços Web que colaborarão com a execução do serviço acompanhados da sua URL de arquivo WSDL e das respectivas operações.

Nesta etapa de preparação o que é importante nesta ontologia são as URLs para os arquivos WSDL dos Serviços Web e as respectivas operações. De posse destas URLs busca-se no Mecanismo de Registro o URL do arquivo WSDL de cada um dos serviços Web. Cada um dos arquivos WSDL contêm informações importantes para a geração automática de proxies de serviços Web, tais como, o nome do serviço, os nomes das operações, os nomes dos parâmetros de cada operação e a URL do local onde efetivamente reside o serviço Web. Os nomes das operações contidos na ontologia determinarão os proxies necessários para invocar as operações nos respectivos serviços Web. Complementando as informações anteriores, estas também são mapeadas na estrutura do Perfil de Serviço. Uma vez recuperados todos os arquivos WSDL pode-se concluir que os Serviços Web necessários para realizar o serviço estão potencialmente disponíveis.

5.2.6 A Verificação da Consistência de Dados Fundamentais

É necessário também a verificação da consistência dos dados fundamentais, que consiste no mapeamento dos dados obtidos até então nos parâmetros dos serviços Web. Por exemplo, considerando-se que seja necessário realizar o cancelamento do RG, o número do RG é um parâmetro fundamental e, portanto, é necessário que entre os dados solicitados ao cidadão esteja o número do RG, que, no caso, para o cidadão "José Silva" é o número "123". A indisponibilidade de algum dado fundamental como parâmetro de entrada de um Serviço Web inviabiliza a realização do serviço para o cidadão, porém, nem todos os serviços Web, como por exemplo os de "registrar um BO" ou "pagar uma fatura", exigem a disponibilidade de todos os dados, como fundamentais, para iniciar a sua execução. Serviços Web como estes podem ser, por si só, serviços ao cidadão e apresentam uma lógica de interação própria e independente de outros serviços.

5.2.7 O Salvamento do Perfil de Serviço

Após estas verificações, o Preparador grava no Repositório de Perfis, o Perfil de Serviço com todos os elementos relevantes para a realização do serviço. A Tabela 5.5 apresenta, como exemplo, um trecho do Perfil de Serviço para o serviço "Solicitar Segunda Via de RG".

Tabela 5.5 Representação XML de um Perfil de Serviço (Solicitação de Segunda Via de RG).

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy by Aqueo Kamada-->
<perfilServico xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="F:\01.Tese\03.Minhas
Propostas\Draft10\XML\Examples\solicitaSegundaViaRG.xsd">
 <nomeServico>SolicitaSegundaViaRG</nomeServico>
 <identServico>SP.SSP.ID.CIVIL.SSVRG.01</identServico>
 <uriDescritor>http://localhost:8080/simpleservice/services/SSVRG.wsdl</uriDescritor>
 <DadosGerais>
    <Parametro>
       <Sequencia>
          <nome>razaoServico</nome> <valor>Perda de RG</valor>
          <nome>nomeCidadao</nomeCidadao> <valor>José Silva</valor>
          <nome>numeroRG</nome> <valor>123</valor>
          <nome>numeroCPF</nome> <valor>987</valor>
          <nome>rua</nome> <valor>Rua Nove, 1</valor>
          <nome>cidade</nomeCidadao> <valor>Campinas</valor>
          <nome>estado</nome> <valor>SP</valor>
          <nome>cep</nome> <valor>13080470</valor>
       </Sequencia>
    </Parametro>
    razaoServico> </razaoServico>
 </DadosGerais>
 <ComponenteServico>
    <nomeServico>RG</nomeServico>
    <identServico>SP.SSP.ID.CIVIL.RG.01</identServico>
    <uriDescritor>http://localhost:8080/simpleservice/services/RG.wsdl</uriDescritor>
    <operacao>
       <nomeOperacao>cancelaRG</nomeOperacao>
       <parametro>
          <Simples>
             <nome>numeroRG</nome> <valor>123</valor>
          </Simples>
       </parametro>
    </operacao>
    <operacao>
       <nomeOperacao>consultaStatusRG</nomeOperacao>
       <parametro>
          <Simples>
             <nome>numeroRG</nome> <valor>123</valor>
          </Simples>
       </parametro>
```

```
    </operacao>
 </ComponenteServico>
 <ComponenteServico>
    <nomeServico>BO</nomeServicob>
    <identServico>SP.SSP.ID.CIVIL.BO.01</identServico>
    <uriDescritor>http://localhost:8080/simpleservice/services/BO.wsdl</uriDescritor>
    <operacao>
        <nomeOperacao>registraBO</nomeOperacao>
        <parametro>
            <Sequencia>
                <nome>numeroBO</nome> <valor/>
                <nome>numeroRG</nome> <valor>123</valor>
                <nome>nomeCidadao</nome> <valor>José Silva</valor>
            </Sequencia>
         </parametro>
    </operacao>
 </ComponenteServico>
 ...
</perfilServico>
```

5.2.8 A Obtenção das Regras de Negócio

O Executor recupera o Perfil de Serviço no Repositório de Perfis e, a partir do nome do serviço ou do identificador do serviço obtém do Repositório de Regras o conjunto de regras associado ao serviço. Esse conjunto de regras inclui as regras que têm relação direta com a necessidade, no caso, *"Solicitar a Segunda Via do RG"*, e com a razão *"perda do RG"* que embasa a necessidade.

5.2.9 A Criação de Programas-Cliente para Serviços Web

Tendo como parâmetros os URLs de arquivos WSDL contidos no Perfil de Serviço, busca-se os respectivos arquivos WSDL para cada um dos Serviços Web necessários para realizar o serviço. Baseado nos arquivos WSDL e nos dados fornecidos pelo cidadão, também contidos no Perfil de Serviço, gera-se os programas-cliente (*proxies)* para os Serviços Web que invocarão as operações nos respectivos Serviços Web.

5.2.10 A Instanciação da Máquina de Regras e a Tradução das Regras

Uma vez preparada a execução do serviço, o Executor instancia a máquina de regras Jess. Em seguida o Executor traduz as regras que foram recuperadas do Repositório de Regras (cujos modelos são aderentes ao metamodelo SBVR) para o modelo da máquina de regras Jess, já com as devidas marcações para as chamadas aos serviços Web. A tradução completa é carregada na memória de trabalho da máquina de regras Jess.

5.2.11 A Execução das Regras e Serviços Web

O Executor executará a máquina de regras considerando o conteúdo da memória de trabalho que neste instante tem todas as regras e fatos necessários para disparar a execução de regras. As regras de negócio, uma vez executadas, gerarão novos fatos, que vão disparar a execução de outras regras. Algumas destas regras terão como ação a chamada de serviços Web, que individualmente realizarão parte do serviço e no conjunto realizarão o serviço desejado pelo cidadão. Por exemplo, na regra de cancelamento de RG, considerando-se que o "cidadão **não** registrou o BO", pelo mecanismo de encadeamento regressivo de Jess a outra regra que trata do registro de BO será disparada e, se a ação realizada pelo correspondente serviço Web for concluída com sucesso, ocorrerá a geração do novo fato "cidadao_registrou_BO". Ato contínuo, a regra anterior, de cancelamento de RG será agora disparada e a ação de cancelamento de RG realizada pelo respectivo serviço Web possibilitará a geração do novo fato "RG_foi_cancelado".

A idéia é que, com as regras de negócio associadas ao desejo e pelo menos um fato associado à razão, o serviço possa ser iniciado. Este fato é que dará início à execução das regras de negócio que dispararão ações que poderão invocar a execução de Serviços Web que por sua vez poderão gerar novos fatos. Estes novos fatos darão início à execução de novas regras de negócio e assim, sucessivamente, novos Serviços Web serão executados, novos fatos serão gerados, até que o desejo do cidadão seja realizado, ou seja, o serviço seja concluído.

5.3 Aspectos da Manutenção dos Repositórios de Regras e de Ontologia

Esta seção descreve alguns aspectos da criação e edição dos elementos tratados pelo Repositório de Regras e Repositório de Ontologia. Conforme dito anteriormente, a manutenção destes elementos é feita através do Editor do IDE.

5.3.1 Os Elementos do Vocabulário

A Tabela 5.6 apresenta alguns exemplos de frases descrevendo necessidades ou eventos de vida relacionados com os elementos do vocabulário da comunidade de identificação civil e que são utilizados pelo Preparador para definir o escopo do serviço.

Tabela 5.6 - Possíveis itens relacionados com as frases de entrada

Exemplos de entradas descrevendo necessidades, incidentes ou insatisfações	**Verbos relacionados com a necessidade**	**Termos e Qualificadores**	**Questões subseqüentes para definir o escopo do serviço**	**Razão ou situação**
Necessidade: Eu **preciso** de uma **nova identidade**. Eu **quero tirar** a **segunda via** da minha **carteira de identidade** (RG).	Quero, quero obter, quero tirar, desejo, preciso, necessito	carteira de identidade, identidade, RG, nova, outra, segunda via	Entre com o número do RG, Nome, Estado Emissor, :..... Por que deseja solicitar a segunda via da carteira de identidade? Respostas: -Porque eu perdi minha carteira de identidade (1). - Porque a minha carteira de identidade foi roubada (2). - Porque a minha carteira de identidade está velha (3).	1- Cidadão perdeu RG, ou RG do cidadão foi furtado. 2- RG do cidadão foi roubado. 3- RG do cidadão está velho, ou rasgado, ou danificado.
Necessidade: Eu **quero obter** uma **carteira de identidade**.	Quero, quero obter, quero tirar, desejo, preciso	carteira de identidade, identidade, RG	É a primeira vez que está solicitando o RG? Respostas: -Sim (4) -Não (5) 5- Entre com o número do RG, Nome, Estado Emissor, :..... -Por que deseja a segunda via da carteira de identidade? Respostas: -Eu perdi minha carteira de identidade (1). -A minha carteira de identidade foi roubada (2). -A minha carteira de identidade está velha (3).	4- Cidadão não tem carteira de identidade 1- Perda da carteira de identidade. 2- Roubo da carteira de identidade. 3- Carteira de identidade está antiga, velha, rasgada, amassada, ou destruída
Evento: Eu **perdi** minha **carteira de identidade.** (1) Eu **perdi** minha **identidade**. (1) Eu **perdi** meu **RG.** (1) Meu **RG** foi **roubado**. (2) Minha **identidade** foi **furtada**. (1)	Perdi, foi perdida, foi roubada, foi furtada	carteira de identidade, identidade, RG	"Por favor, entre com os dados de seu RG (**numero**, **estado emissor**, e se possível a data de emissão):....." Deseja solicitar a segunda via da carteira de identidade? -Sim (desejo) -Não (6) 6- Já registrou um BO? -Sim (8) -Não (9) 9- Você precisa registrar um BO urgentemente para que seu RG possa ser cancelado e evitar qualquer uso indevido! Deseja registrar um BO neste momento? -Sim (10) **(necessidade)**	1- cidadão perdeu RG ou cidadão teve seu RG furtado 2- cidadão teve seu RG roubado 8- cidadão registrou BO 11- cidadão em situação de risco

			-Não (11) 11- Você está em situação de risco!.	

5.3.2 A Definição de Termos

Estes termos, verbos, qualificadores e questões são muito familiares às pessoas da comunidade de identificação civil, de modo que a inclusão destes elementos, através do editor do IDE, é bastante trivial e de entendimento imediato para essas pessoas. A Figura 5.2 mostra o protótipo de interface do IDE com o Editor de Termos ativado para a inserção de termos. De modo similar edita-se verbos, qualificadores e questões.

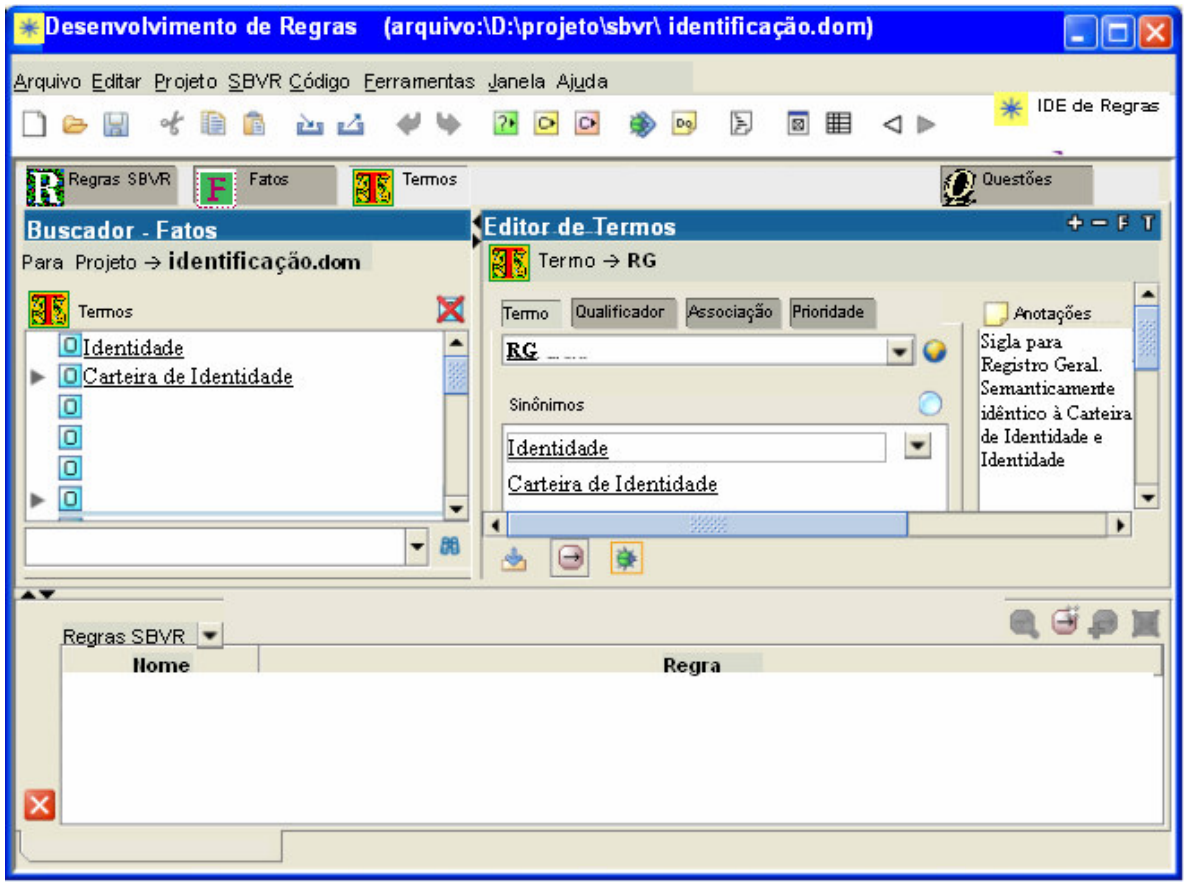

Figura 5.2 - Editor de Termos ativado.

No instante de definição destes elementos o Editor deve criar a representação para eles no vocabulário da comunidade. Assim, elementos tais como os contidos na Tabela 5.6 podem ser representados no Repositório de Regras conforme o modelo aderente ao padrão UML/MOF, mostrado na Figura 5.3.

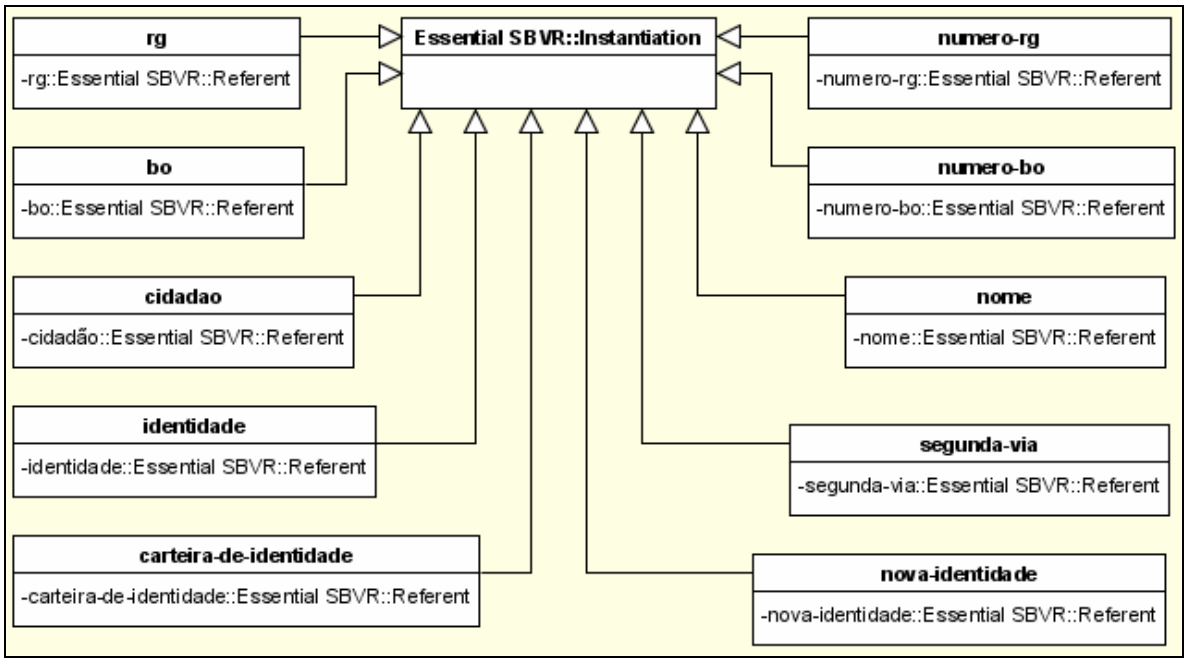

Figura 5.3 - Modelo UML/MOF para parte do vocabulário de identificação civil.

Quando, no Editor, indica-se que o termo "RG" tem como sinônimos "Identidade" e "Carteira de Identidade" o Editor cria fatos automaticamente usando um modelo interno na forma de "é-sinônimo-de-". Estes fatos são acrescentados no Repositório de Regras conforme representação mostrada no diagrama da Figura 5-4.

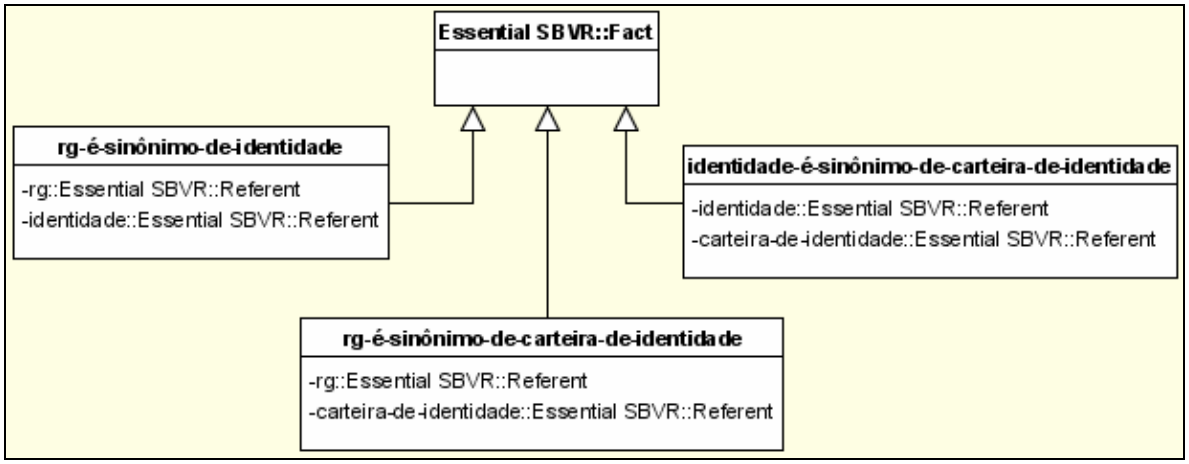

Figura 5.4 - Fatos para representar sinônimos.

5.3.3 A Definição de Fatos

Além dos fatos que são criado automaticamente há os fatos que comporão as condições e ações das regras. Estes fatos poderão ser definidos isoladamente ou no instante de criação de regras. A Figura 5.5 mostra a seleção de modelo de fatos para definir um fato isoladamente e a Figura 5.6 mostra a mesma interface do Editor com alguns fatos já definidos.

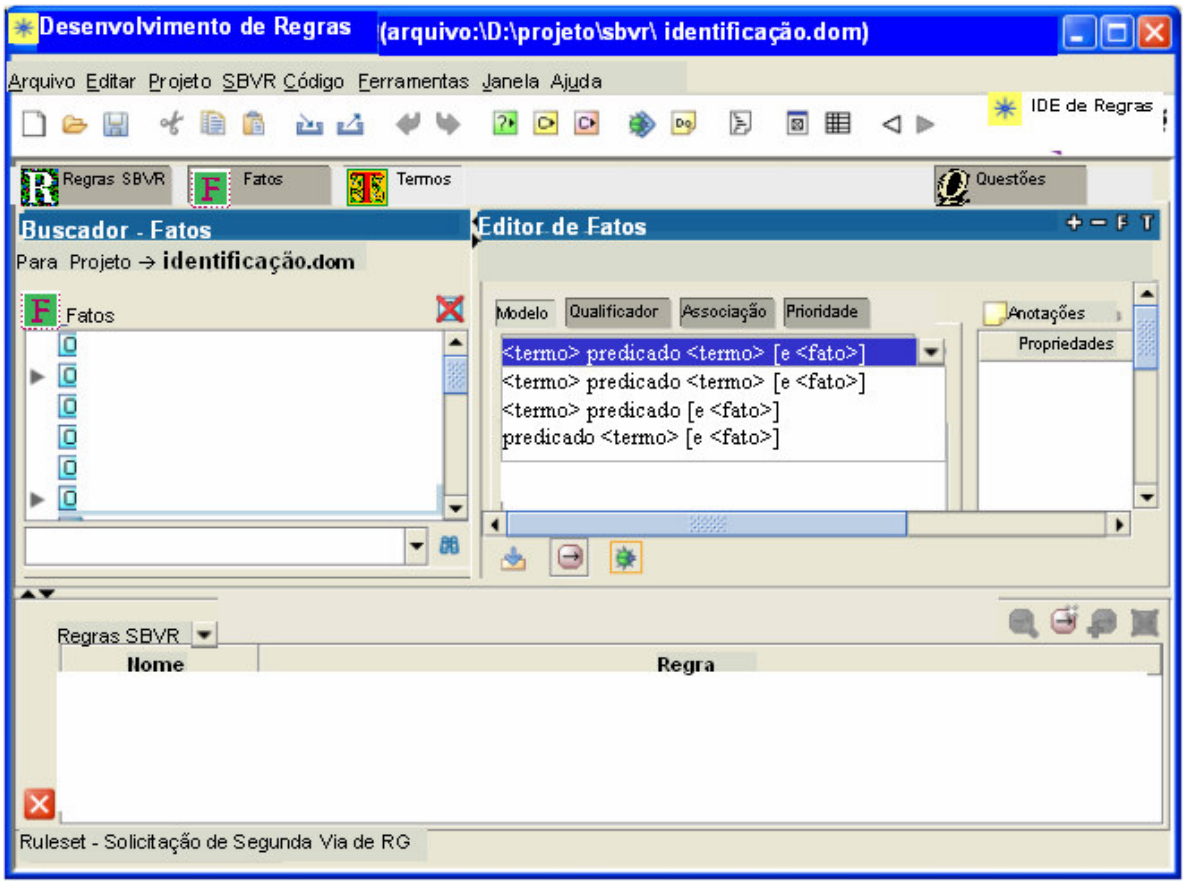

Figura 5.5 - Editor de Fatos – Seleção de *template*.

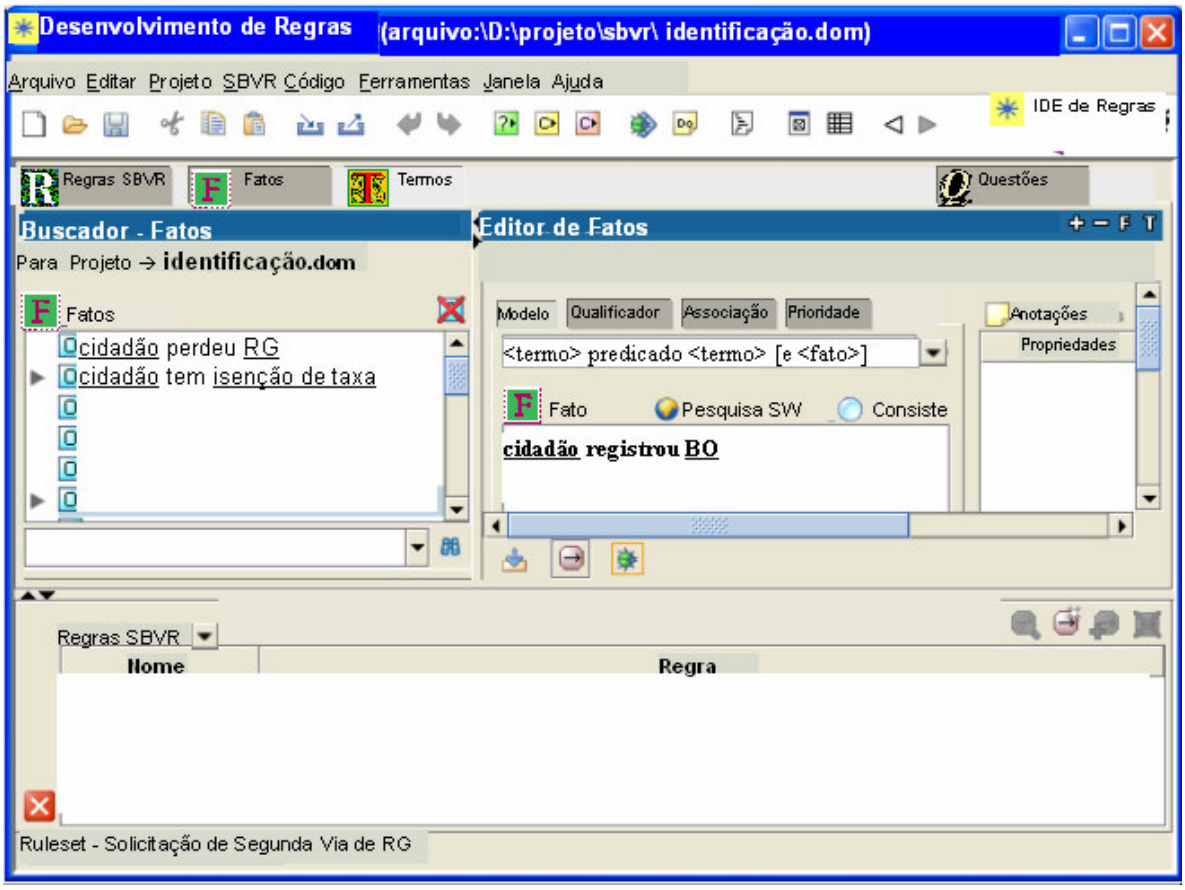

Figura 5.6 - Editor de Fatos - Fatos definidos.

Neste ponto, é possível pesquisar por Serviços Web que realizem eventuais ações embutidas no fato em definição. Um clique no ícone "Pesquisa SW" e o Editor pesquisará no Mecanismo de Registro por URL de serviço Web para fazer o registro de Boletim de Ocorrência. Estes fatos são também acrescentados no Repositório de Regras conforme representação mostrada na Figura 5-7.

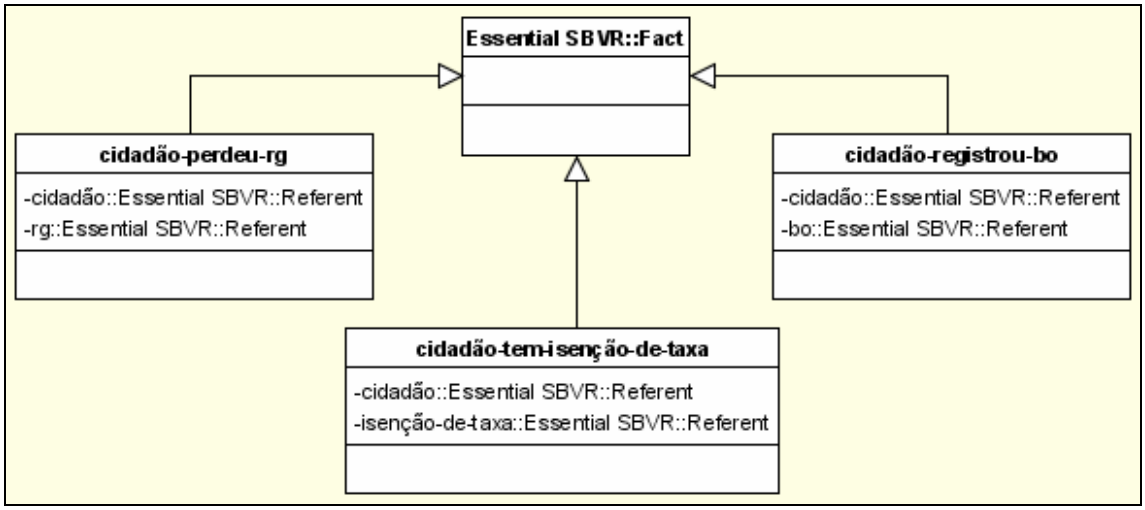

Figura 5.7 - Novos Fatos acrescentados pelo IDE.

Estes modelos em notação UML/MOF do sub-conjunto do vocabulário de identificação civil pode ser representado pelo trecho XML Schema mostrado na Tabela 5.7.

Tabela 5.7 - XML Schema do modelo UML/MOF para o vocabulário parcial de identificação civil.

```
<xsd:complexType name="bo" >
        <xsd:complexContent >
                <xsd:extension base="esbvr:Instantiation">
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element name="bo" type="esbvr:Referent"/>
                        </xsd:choice>
                        <xsd:attribute name="branch" type="xsd:IDREF" use="optional"/>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
<xsd:element name="bo" type="bo"/>

<xsd:complexType name="rg" >
        <xsd:complexContent >
                <xsd:extension base="esbvr:Instantiation">
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element name="rg" type="esbvr:Referent"/>
                        </xsd:choice>
                        <xsd:attribute name="car" type="xsd:IDREF" use="optional"/>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
<xsd:element name="rg" type="rg"/>

<xsd:complexType name="cidadao" >
        <xsd:complexContent >
                <xsd:extension base="esbvr:Instantiation">
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
```

```
                                    <xsd:element name="cidadao" type="esbvr:Referent"/>
                            </xsd:choice>
                            <xsd:attribute name="car" type="xsd:IDREF" use="optional"/>
                    </xsd:extension>
            </xsd:complexContent>
</xsd:complexType>
<xsd:element name="cidadao" type="cidadao"/>

<xsd:complexType name="nome" >
            <xsd:complexContent >
                    <xsd:extension base="esbvr:Instantiation">
                            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                    <xsd:element name="nome" type="esbvr:Referent"/>
                            </xsd:choice>
                            <xsd:attribute name="car" type="xsd:IDREF" use="optional"/>
                    </xsd:extension>
            </xsd:complexContent>
</xsd:complexType>
<xsd:element name="nome" type="nome"/>

<xsd:complexType name="numero-rg" >
            <xsd:complexContent >
                    <xsd:extension base="esbvr:Instantiation">
                            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                    <xsd:element name="numero-rg" type="esbvr:Referent"/>
                            </xsd:choice>
                            <xsd:attribute name="car" type="xsd:IDREF" use="optional"/>
                    </xsd:extension>
            </xsd:complexContent>
</xsd:complexType>
<xsd:element name="numero-rg" type="numero-rg"/>

<xsd:complexType name="numero-bo" >
            <xsd:complexContent >
                    <xsd:extension base="esbvr:Instantiation">
                            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                    <xsd:element name="numero-bo" type="esbvr:Referent"/>
                            </xsd:choice>
                            <xsd:attribute name="car" type="xsd:IDREF" use="optional"/>
                    </xsd:extension>
            </xsd:complexContent>
</xsd:complexType>
<xsd:element name="numero-bo" type="numero-bo"/>

<xsd:complexType name="cidadao-tem-nome" >
            <xsd:complexContent >
                    <xsd:extension base="esbvr:Fact">
                            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                    <xsd:element name="cidadao" type="esbvr:Referent"/>
                                    <xsd:element name="nome" type="esbvr:Referent"/>
                            </xsd:choice>
                            <xsd:attribute name="cidadao" type="xsd:IDREF" use="optional"/>
                            <xsd:attribute name="nome" type="xsd:IDREF" use="optional"/>
                    </xsd:extension>
```

```
        </xsd:complexContent>
</xsd:complexType>
<xsd:element name="cidadao-tem-nome" type="cidadao-tem-nome"/>

<xsd:complexType name="cidadao-tem-rg" >
        <xsd:complexContent >
                <xsd:extension base="esbvr:Fact">
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element name="cidadao" type="esbvr:Referent"/>
                                <xsd:element name="rg" type="esbvr:Referent"/>
                        </xsd:choice>
                        <xsd:attribute name="cidado" type="xsd:IDREF" use="optional"/>
                        <xsd:attribute name="rg" type="xsd:IDREF" use="optional"/>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
<xsd:element name="cidadao-tem-rg" type="cidadao-tem-rg"/>

<xsd:complexType name="cidadao-perdeu-rg-e-registrou-bo" >
        <xsd:complexContent >
                <xsd:extension base="esbvr:Fact">
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                            <xsd:element name="cidadao" type="esbvr:Referent"/>
                            <xsd:element name="rg" type="esbvr:Referent"/>
                            <xsd:element name="bo" type="esbvr:Referent"/>
                        </xsd:choice>
                        <xsd:attribute name="car" type="xsd:IDREF" use="optional"/>
                        <xsd:attribute name="vehicle-identification-number" type="xsd:IDREF"
                          use="optional"/>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
<xsd:element name="cidadao-perdeu-rg-e-registrou-bo"
    type="cidadao-perdeu-rg-e-registrou-bo"/>
```

O trecho da Tabela 5.8 mostra como os fatos dos diagramas anteriores podem ser representados e validados pelo XML Schema produzido a partir do modelo UML/MOF, que por sua vez foi baseado no metamodelo SBVR.

Tabela 5.8 - Exemplo de fatos validados pelo XML Schema.

```
<bo-tem-numero-bo bo="b" numero-bo="x"/>
<cidadão-tem-nome cidadão="c" nome="n"/>
<cidadão-tem-rg cidadão="c" rg="r" />
<rg-tem-numero-rg rg="r" numero-rg="y"/>
<cidadão-perdeu-rg-e-registrou-bo cidadão="c" rg="r" bo ="b"/>
<rg-esta-cancelado rg="r"/>
<esbvr:Thing xmi:id="b"/>
<esbvr:Thing xmi:id="c"/>
<esbvr:Thing xmi:id="r"/>
<esbvr:Text xmi:id="x">123</esbvr:Text>
<esbvr:Text xmi:id="n">José Silva</esbvrr:Text>
<esbvr:Text xmi:id="y">987</esbvr:Text>
```

5.3.4 A Definição de Regras de Negócio

Na seqüência será apresentado como ocorre a criação de regras e como os fatos que compõem as regras serão também criados. A Figura 5.8 mostra como um modelo de regra é selecionado.

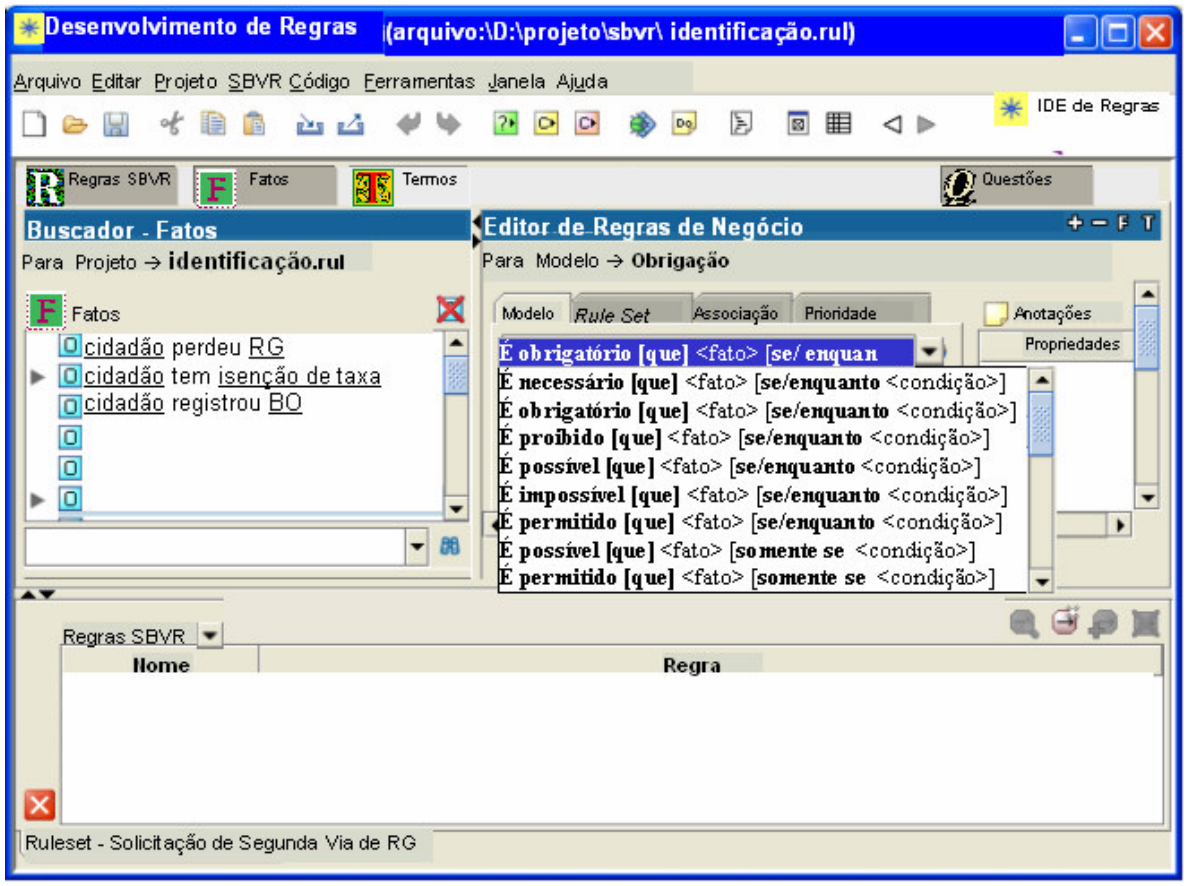

Figura 5.8 - Editor de Regras - Seleção de *template*.

Selecionando-se um modelo e clicando-se sobre <fato> ou <condição> no modelo de edição o Editor busca a lista de predicados possíveis naquele contexto, conforme mostra a Figura 5.9.

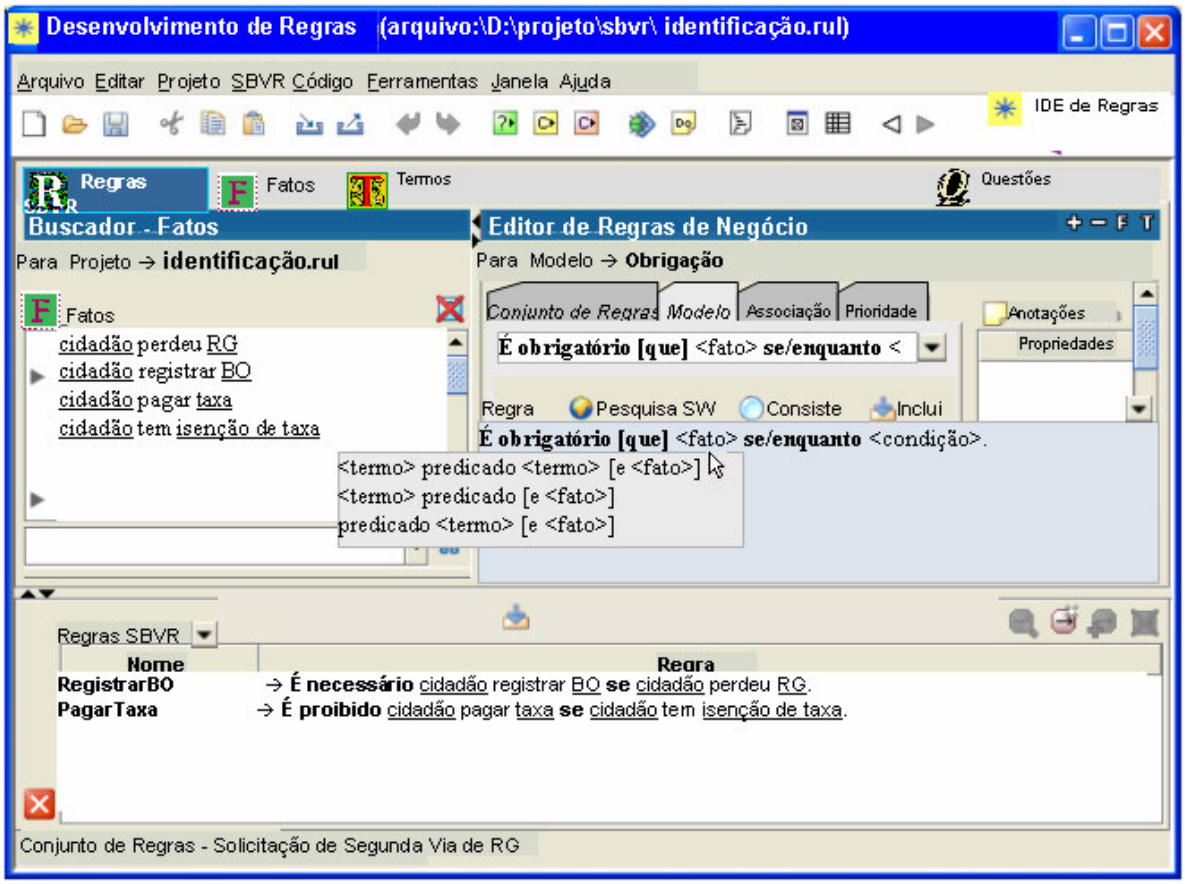

Figura 5.9 - Editor de Regras - Seleção de Predicado.

Selecionando-se um predicado o modelo ficará expandido e clicando-se sobre um <termo> do predicado o Editor buscará a lista de termos possíveis. Outra alternativa para incluir fatos num modelo de regra é selecionar um fato no quadro à esquerda do Editor e arrastá-lo para a posição <fato> ou <condição> no modelo. Uma vez completada a definição de uma regra a sua inclusão no conjunto de regras é feita através do botão Inclui no conjunto de regras. A Figura 5.10 mostra algumas regras criadas e gravadas no Repositório de Regras.

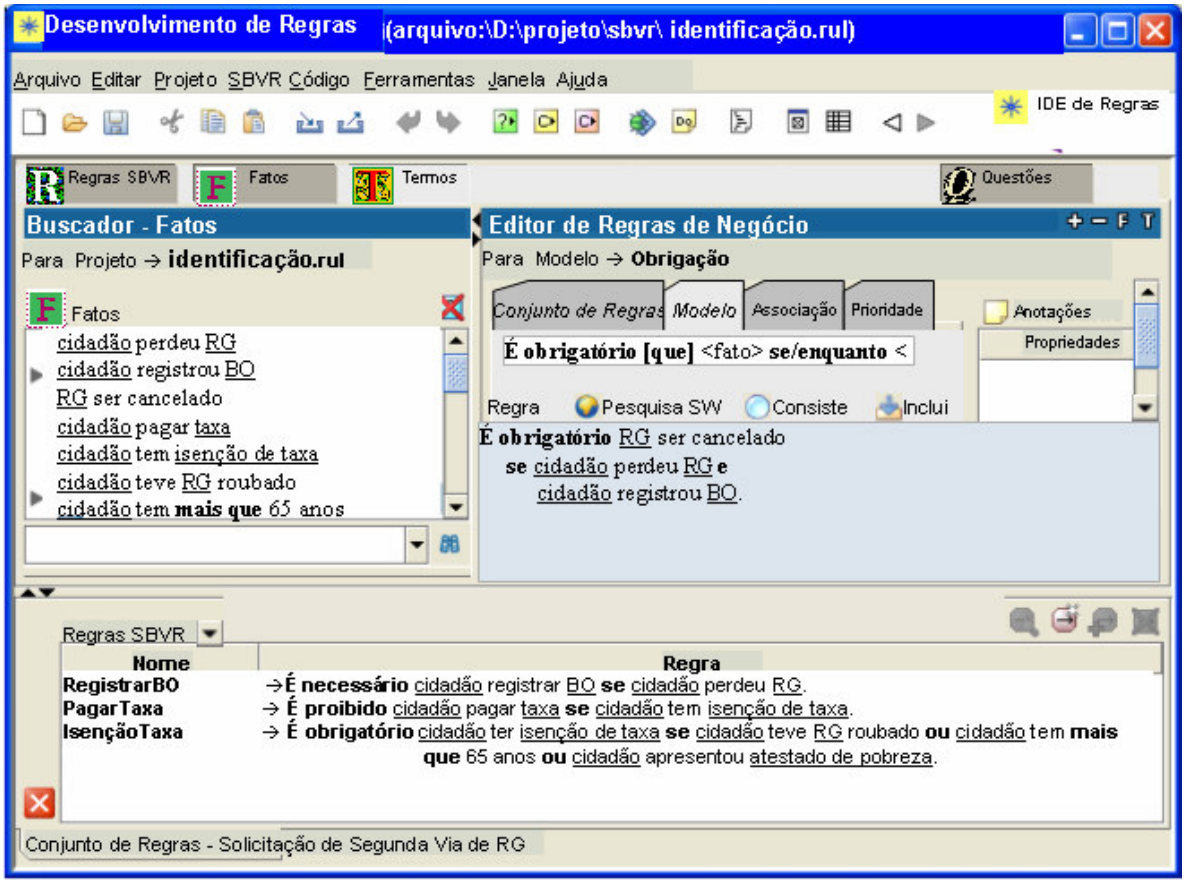

Figura 5.10 - Editor de Regras - Regras definidas.

Os eventuais termos, verbos ou predicados ainda não definidos quando da definição de uma regra serão incluídos no vocabulário em questão, no Repositório de Regras. Para esta regra que trata do cancelamento de RG, recém definida, é acrescentada no Repositório de Regras uma representação como a mostrada na Figura 5.11.

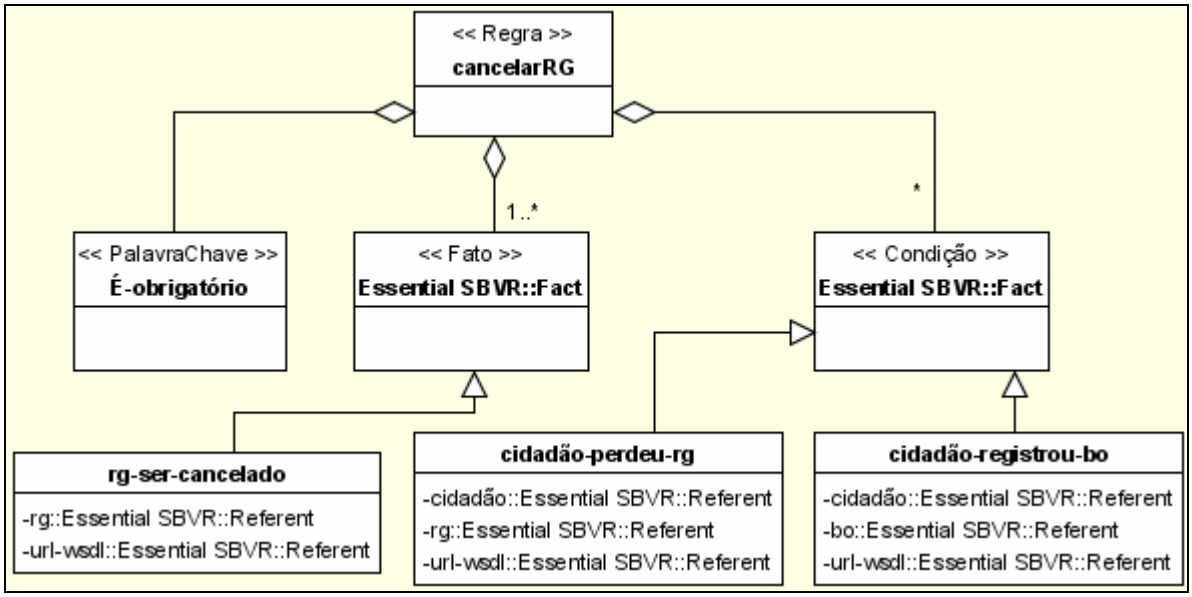

Figura 5.11 - Modelo para a regra de cancelamento de RG.

5.3.5 A Ligação de Serviços Web na Ontologia de Serviço

Uma vez definida uma regra de negócio é possível pesquisar por Serviços Web para realizar eventuais ações embutidas nos fatos que compõem a regra. Um clique no ícone "Pesquisa SW" e o Editor pesquisará no Mecanismo de Registro por URLs de Serviços Web para fazer o cancelamento de RG e para fazer o registro de Boletim de Ocorrência. Para o fato "cidadão perdeu RG" o Editor não pesquisará porque é um fato associado a evento de vida. Obviamente que, mesmo que a pesquisa fosse disparada o Mecanismo de Registro não encontraria nenhum URL de serviço Web para "realizar a perda de RG". Para cada URL encontrado busca-se o arquivo WSDL, obtém-se dele os nomes das operações que são incluídos na estrutura da ontologia para o serviço no Repositório de Ontologia.

Para mostrar a estrutura ontológica de Serviços Web considerar, por exemplo, o serviço "Solicitar Segunda Via de RG", que comportaria as informações mostradas na Tabela 5.9.

Tabela 5.9 - Ontologia de Serviços Web.

Ontologia Nome da ontologia: Identificação Civil - RG

```
Nome da operação: solicitar segunda via de RG
Nome do serviço Web: RG
URL do serviço Web: htth://www.sp.gov.br/RG/RG.wsdl
    Propriedade: (versão, "V2006-10-22")
    Propriedade: (descrição, "Esta ontologia estrutura o serviço para solicitar segunda via de
                RG, no contexto de identificação civil")
    Propriedade: (autor, "Aqueo Kamada")
    ComponenteServiço
      Nome do serviço Web: RG
      URL do serviço Web: ://www.sp.gov.br/RG/RG.wsdl
          Operação: cancelarRG
          Operação: cadastrarRG
          Operação: consultarRG
    ComponenteServiço
      Nome do serviço Web: BO
      URL do serviço Web: ://www.sp.gov.br/BO/BO.wsdl
          Operação: registrarBO
          Operação: consultarBO
    ComponenteServiço
      Nome do serviço Web: RGPT
      URL do serviço Web: www.sp.gov.br/RGPT/RGPT.wsdl
          Operação: emitirSegundaViaRG
          Operação: agendarSegundaViaRG
    ComponenteServiço
      Nome do serviço Web: Criminal
      URL do serviço Web: http://www.sp.gov.br/Criminal/Criminal.wsdl
          Operação: consultarFichaCriminal
    ComponenteServiço
      Nome do serviço Web: CAP
      URL do serviço Web: http://www.sp.gov.br/CAP/CAP.wsdl
          Operação: consultarImpressoesDigitais
    ComponenteServiço
      Nome do serviço Web: BP_Recebimento
      URL do serviço Web: http://www.abc.com.br/BP /BP_Recebimento.wsdl
          Operação: receberTaxaSegundaViaRG
```

A ontologia de serviços é estruturada criando-se representações semelhantes a esta para todos os serviços de uma comunidade de negócios, na linguagem de representação do repositório. Considerando-se que OWL é uma linguagem bastante utilizada para estruturar ontologias (com disponibilização de ferramentas populares como Protégé OWL [130] e Jena 2 [131]) o trecho de código OWL mostrado na Tabela 5.10 ilustra dois desses relacionamentos, especificamente a relação entre o serviço e os Serviços Web "Registar BO" e "Cancelar RG".

Tabela 5.10 - Trecho de ontologia descrita em OWL.

```
...
<owl:Class rdf:ID="solicitarSegundaViaRG ">
        <rdfs:comment>Solicitar Segunda Via de RG.</rdfs:comment>
        <rdfs:subClassOf rdf:resource="#serviço"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="identificadorServiço">
   <owl:equivalentProperty rdf:resource="#identificadorServiçoWeb"/>
   <rdfs:domain rdf:resource="#serviço"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="registrarBO">
        <rdfs:comment>Serviço Web para Registrar BO na solicitação de segunda via de RG por perda,
        roubo ou furto do RG.</rdfs:comment>
        <rdfs:subClassOf rdf:resource="#serviço"/>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#isPartOf"/>
                        <ow:allValuesFrom rdf:resource="#solicitarSegundaViaRG"/>
                </owl:Restriction>
        </rdfs:subClassOf>
        </owl:Class>
<owl:Class rdf

<owl:Class rdf:ID="cancelarRG">
        <rdfs:comment> Serviço Web para Cancelar RG na solicitação de segunda via de RG por perda,
        roubo ou furto do RG.</rdfs:comment>
        <rdfs:subClassOf rdf:resource="#serviço"/>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#isPartOf"/>
                        <ow:allValuesFrom rdf:resource="#solicitarSegundaViaRG"/>
                </owl:Restriction>
        </rdfs:subClassOf>
        </owl:Class>
<owl:Class rdf
…
```

5.4 Aspectos de Preparação e Execução do Serviço

Nesta seção discutem-se alguns aspectos da preparação e da execução do serviço, desde a verificação de disponibilidade de Serviços Web até a finalização do serviço.

Para verificar a disponibilidade dos serviços Web, o Preparador pesquisa o Repositório de Ontologia e obtém a ontologia que refere-se ao serviço. Esta ontologia contém os identificadores dos Serviços Web associados, como por exemplo, <owl:Class

rdf:ID="registrarBO">. De posse destes identificadores e outros apontadores típicos da comunidade ou da organização busca-se no Mecanismo de Registro o URL do arquivo WSDL de cada um dos serviços Web. Com estes URLs obtém-se os respectivos arquivos WSDL. O resultado desta busca pode ser consolidado em arquivos WSDL semelhantes ao mostrado na Tabela 5.11, para operações sobre RG.

Tabela 5.11 - Arquivo WSDL para serviço Web com operações sobre RG.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://proj_rg3"
xmlns:intf="http://proj_rg3" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://proj_rg3">
  <wsdl:message name="setCadastroRGRequest">
    <wsdl:part name="cadastroRG" type="xsd:boolean"/>
  </wsdl:message>
  <wsdl:message name="consultaStatusRGResponse">
    <wsdl:part name="consultaStatusRGReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="setCadastroRGResponse">
 </wsdl:message>
  <wsdl:message name="cancelaRGRequest">
    <wsdl:part name="numeroRG" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="consultaCadastroRGRequest">
 </wsdl:message>
  <wsdl:message name="consultaCadastroRGResponse">
    <wsdl:part name="consultaCadastroRGReturn" type="xsd:boolean"/>
  </wsdl:message>
  <wsdl:message name="consultaStatusRGRequest">
 </wsdl:message>
  <wsdl:message name="cancelaRGResponse">
    <wsdl:part name="cancelaRGReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:portType name="RG">
    <wsdl:operation name="consultaStatusRG">
        <wsdl:input name="consultaStatusRGRequest" message="impl:consultaStatusRGRequest"/>
        <wsdl:output name="consultaStatusRGResponse" message="impl:consultaStatusRGResponse"/>
    </wsdl:operation>
    <wsdl:operation name="cancelaRG" parameterOrder="numeroRG">
        <wsdl:input name="cancelaRGRequest" message="impl:cancelaRGRequest"/>
        <wsdl:output name="cancelaRGResponse" message="impl:cancelaRGResponse"/>
    </wsdl:operation>
    <wsdl:operation name="consultaCadastroRG">
        <wsdl:input name="consultaCadastroRGRequest" message="impl:consultaCadastroRGRequest"/>
        <wsdl:output name="consultaCadastroRGResponse"
message="impl:consultaCadastroRGResponse"/>
    </wsdl:operation>
    <wsdl:operation name="setCadastroRG" parameterOrder="estaRegistrado">
        <wsdl:input name="setCadastroRGRequest" message="impl:setCadastroRGRequest"/>
        <wsdl:output name="setCadastroRGResponse" message="impl:setCadastroRGResponse"/>
    </wsdl:operation>
```

```
  </wsdl:portType>
  <wsdl:binding name="RGSoapBinding" type="impl:RG">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="consultaStatusRG">
        <wsdlsoap:operation/>
        <wsdl:input>
          <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://proj_rg3"/>
        </wsdl:input>
        <wsdl:output>
          <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://proj_rg3"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="cancelaRG">
        <wsdlsoap:operation/>
        <wsdl:input>
          <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://proj_rg3"/>
        </wsdl:input>
        <wsdl:output>
          <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://proj_rg3"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="consultaCadastroRG">
        <wsdlsoap:operation/>
        <wsdl:input>
          <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://proj_rg3"/>
        </wsdl:input>
        <wsdl:output>
          <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://proj_rg3"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="setCadastroRG">
        <wsdlsoap:operation/>
        <wsdl:input>
          <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://proj_rg3"/>
        </wsdl:input>
        <wsdl:output>
          <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://proj_rg3"/>
        </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="RG">
    <wsdl:port name="RG" binding="impl:RGSoapBinding">
        <wsdlsoap:address location="http://localhost:8080/simpleservice/services/RG"/>
    </wsdl:port>
  </wsdl:service>
  <!--WSDL created by Apache Axis version: 1.2.1
```

```
Built on Aug 08, 2005 (11:49:10 PDT)-->
</wsdl:definitions>
```

Cada um dos arquivos WSDL contêm informações importantes, tais como, nome do serviço, nomes das operações, nomes dos parâmetros de cada operação e URL do local onde efetivamente reside o serviço Web.

5.4.1 O Perfil de Serviço

O trecho em XML Schema do Perfil de Serviço está codificado na Tabela 5.12. Esse XML Schema pode ser estruturado num SGBD convencional, relacional ou orientado a objetos, ou mais apropriadamente, num SGBD XML, tal como, eXist [132], Xindice [133] e Berkeley DB XML [134].

Tabela 5.12 - XML Schema do Perfil de Serviço.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--W3C Schema generated by XMLSpy v2006 rel. 3 sp1 (http://www.altova.com) -->
<!--XML Schema gerado por XMLSpy v2006 rel. 3 sp1 e editado por Aqueo Kamada -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:annotation>
  <xs:documentation xml:lang="en">
        Esquema Perfil de Serviço.
  </xs:documentation>
  </xs:annotation>

  <xs:element name="perfilServico">
      <xs:complexType>
            <xs:sequence>
                <xs:element ref="nomeServico"/>
                <xs:element ref="identServico"/>
                <xs:element ref="uriDescritor"/>
                <xs:element ref="DadosGerais" maxOccurs="unbounded"/>
                <xs:element ref="ComponenteServico" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
  </xs:element>

  <xs:element name="DadosGerais">
      <xs:complexType>
            <xs:sequence>
                <xs:element ref="Parametro" maxOccurs="unbounded"/>
            </xs:sequence>
```

```
        </xs:complexType>
    </xs:element>

    <xs:element name="Parametro">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="nome"/>
                <xs:element ref="valor"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="ComponenteServico">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="nomeServico"/>
                <xs:element ref="identServico"/>
                <xs:element ref="uriDescritor"/>
                <xs:element ref="operacao" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="operacao">
        <xs:complexType>
            <xs:attribute name="nomeOperacao" use="required">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:sequence>
                <xs:element ref="Parametro" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="identServico" type="xs:string"/>
    <xs:element name="nomeServico" type="xs:string"/>
    <xs:element name="nomeOperacao" type="xs:string"/>
    <xs:element name="nome" type="xs:string"/>
    <xs:element name="valor" type="xs:string"/>
    <xs:element name="uriDescritor">
        <xs:simpleType>
            <xs:restriction base="xs:anyURI">
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
</xs:schema>
```

5.4.2 A Geração de Proxies para os Serviços Web

Tendo como parâmetros os URLs de arquivos WSDL contidos nas *tags* "<uriDescritor>" no Perfil de Serviço, busca-se os respectivos arquivos WSDL para cada um dos Serviços Web necessários para realizar o serviço. A Tabela 5.11 apresenta um trecho de arquivo WSDL para a operação "cancelarRG" serviço Web RG.

Baseado nos arquivos WSDL e nos dados fornecidos pelo cidadão geram-se os *proxies* para os serviços Web, ou seja, geram-se os programas-cliente que invocarão as operações nos respectivos Serviços Web. A Tabela 5.13 apresenta a geração de um trecho de código Java para o do serviço Web RG invocando a operação "cancelarRG". O URL atribuído à variável "*endpoint*" foi obtido da *tag* <wsdlsoap:address location> do arquivo WSDL, assim como, os nomes da operação "cancelaRG" e do parâmetro "numeroRG". O valor do parâmetro "numeroRG" foi obtido como entrada, através do Perfil de Serviço.

Tabela 5.13 - *Proxy* do Serviço Web RG para Invocar "cancelaRG".

```
package proj_rg4client;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import javax.xml.namespace.QName;

/**
 * <p>Title: </p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: </p>
 * @author not attributable
 * @version 1.0
 */
public class ProxyRG {
 public ProxyRG () {
 }

 public void cancelaRG(String[] args) {
   try {
    String endpoint = null;
    try {
      endpoint = "http://localhost:8080/simpleservice/services/RG";// location no WSDL
      } catch (Exception ex) {
    }

    Service service = new Service();
    Call call = (Call) service.createCall();

    call.setTargetEndpointAddress(new java.net.URL(endpoint));
    String numeroRG = "987";
    Object[] inputParams = new Object[1];
    inputParams[0] = (Object) numeroRG;

    call.setOperationName(new QName("http://proj_rg3", "cancelaRG"));
    Object resp =  (Object) call.invoke(inputParams);
    System.out.println("Nova Situação do RG = " + resp);

    } catch (Exception e) {
       System.err.println(e.toString());
       }
 }
 public static void main(String[] args) {
 }
```

```
}
```

5.4.3 Considerações sobre Modelagem de Serviços Web

Há situações em que há uma grande variabilidade nos dados de contexto que influenciam o comportamento no instante de execução do serviço. Por exemplo, no instante de realização do serviço de "solicitação de Segunda Via de RG, por perda", várias situações já podem ter ocorrido, tais como, um BO já fora registrado; o RG já fora cancelado por outras razões e o cidadão esqueceu-se; um BO fora registrado, mas o RG não foi cancelado; o RG já foi cancelado, mas não foi solicitada a Segunda Via; etc. Um tratamento adequado dessas variáveis pode fazer diferença em termos de eficiência e desempenho no gerenciamento de fatos derivados e Serviços Web relacionados. Para garantir homogeneidade no uso e tratamento dos Serviços Web pode-se criar um modelo onde define-se que toda operação de serviço Web, que retorne um estado passível de conduzir a novos fatos, deve ter embutida uma verificação para este estado antes da execução da operação.

Assim, na operação "cancelaRG" do serviço Web "RG" seria oportuno verificar se o RG para aquele caso já não fora cancelado, antes de se iniciar efetivamente os procedimentos de cancelar o RG. A Tabela 5.14 apresenta uma adaptação do serviço Web RG para esse modelo enfatizando somente a operação "cancelaRG".

Tabela 5.14 - Modelo de Serviço Web configurado para RG.

```
package proj_rg3;

/**
 * <p>Title: </p>
 *
 * <p>Description: </p>
 *
 * <p>Copyright: Copyright (c) 2006</p>
 *
 * <p>Company: </p>
 *
 * @author not attributable
 * @version 1.0
 */
public class RG {
    String numeroRG;
    String status;
    boolean estaRegistrado;
    //...
```

```
    public RG() {
    }

    public String consultaStatusRG() {
        return status;
    }

    public String cancelarRG(String numeroRG) {
        // Chama sistema legado e verifica se RG já não está cancelado.
        // ...
        if (this.consultaStatusRG() == "Cancelado") {
           return status;
           }
        else {
           // Chama sistema legado para marcar RG como cancelado.
           // ...
           if (this.numeroRG.contentEquals(numeroRG)) {
            status = "Cancelado";
        }
    }

    public boolean consultaCadastroRG() {
        return estaRegistrado;
    }

    public void setCadastroRG(boolean estaRegistrado) {
        this.estaRegistrado = estaRegistrado;
    }
}
```

Este modelo de serviço Web traz a vantagem de minimizar o número de *proxies* gerados e respectivas chamadas, porém, ao cometer a "ingerência" de forçar um modelo reduz a liberdade na criação e a independência no desenvolvimento de serviços Web, além de tornar mais complexa a descrição e, conseqüentemente, o seu uso.

Outra possibilidade é a criação de Serviços Web específicos para fazer esta verificação de estado e criar uma função Jess para chamar todos os Serviços Web que fazem verificação de estado e gerar os respectivos fatos antes de realizar as operações propriamente ditas. Para isto o Executor teria um módulo para gerar a função Jess, tal como a mostrada na Tabela 5.15. Esta função terá uma série de chamadas a Serviços Web associados às precondições das regras, para verificar estados de objetos reais, tais como BO registrado, RG cancelado, e, caso eles existam, gerar os respectivos fatos na memória Jess, através de comandos *assert*.

Tabela 5.15 - Função Jess para verificação de estados de objetos reais.

```
(deffunction VerificaEstadoPreCondicaoRegra
   ; Chama proxy associado à pré-condição cidadaoRegistrouBO para invocar SW
   (defclass sProxyBO ProxyBO)
   (bind ?proxyBO (new ProxyBO))
```

```
    (definstance sProxyBO ?proxyBO static)
    (bind ?numeroBO (call ?proxyBO consultarBO ?nomeCidadao ?numeroRG))
    (if (<> ?numeroBO nil) then
        (assert(cidadao_registrou_BO ?nomeCidadao ?numeroRG ?numeroBO))
    )

    ; Chama proxy associado à pré-condição RGEstaCadastrado para invocar SW
    (defclass sProxyRG  ProxyRG)
    (bind ?proxyRG (new ProxyRG))
    (definstance sProxyRG ?proxyRG static)
    (bind ?ok (call ?proxyRG consultaCadastroRG ?numeroRG))
    (if (= ?ok TRUE) then
        (assert(RGEstaCadastrado ?numeroRG))
    )

    ; Chama proxy associado à pré-condição RGEstaCancelado para invocar SW
    (defclass sProxyRG  ProxyRG)
    (bind ?proxyRG (new ProxyRG))
    (definstance sProxyRG ?proxyRG static)
    (bind ?status (call ?proxyRG consultaStatusRG ?numeroRG))
    (if (= ?status "cancelado") then
        (assert (RGEstaCancelado ?numeroRG))
    )
)
```

Esta alternativa, embora apresente a desvantagem de exigir a geração de um número maior de *proxies* e da geração da função Jess para chamar estes proxies, tem a vantagem fazer a asserção de possíveis fatos com antecedência e evitar-se algumas chamadas de serviços Web, além de dar mais liberdade na criação de Serviços Web e simplificar suas descrições e seus usos.

5.4.4 A Tradução de Regras

Uma vez preparada a execução do serviço, o Executor instancia a máquina de regras Jess. Em seguida o Executor traduz as regras que foram recuperadas do Repositório de Regras (cujos modelos são aderentes ao metamodelo SBVR) para o modelo da máquina de regras Jess, já com as devidas marcações para as chamadas aos serviços Web. A tradução completa é carregada na memória de trabalho da máquina de regras Jess. A Tabela 5.16 exemplifica como fica a tradução de uma regra de negócio que inclui uma cláusula de obrigação e, portanto em Jess necessita valer-se do uso do mecanismo de encadeamento

regressivo (*backward-chaining*). Para simplificar, considera-se que a obrigatoriedade recaia somente sobre o registro do BO.

Tabela 5.16 - Exemplo Tradução de Regras (SBVR-Jess).

Regra em notação aderente ao metamodelo SBVR:

É obrigatório RG ser cancelado **somente se todos os seguintes fatos forem verdade:**

- cidadão perdeu RG
- cidadão registrou BO
- RG está cadastrado

Equivalente tradução em Jess:

```
(do-backward-chaining cidadao_registrou_BO)
(defrule Cancelar_RG
    (cidadao_perdeu_RG ?nomeCidadao ?numeroRG)
    (cidadao_registrou_BO ?nomeCidadao ?numeroRG ?numeroBO)
    (RG_esta_cadastrado ?nomeCidadao ?numeroRG)
    =>
   (defclass sProxyRG  ProxyRG)
   (bind ?proxyRG (new ProxyRG))
   (definstance sProxyRG ?proxyRG static)
    (if (call ?proxyRG cancelarRG ?numeroRG) then
          (assert (RGEstaCancelado ?numeroRG))
    )
)

(defrule find-cidadao_registrou_BO
    (need-cidadao_registrou_BO ?nomeCidadao ?numeroRG ?)
    =>
   (defclass sProxyBO ProxyBO)
   (bind ?proxyBO (new ProxyBO))
   (definstance sProxyBO ?proxyBO static)
   (bind ?numeroBO (call ?proxyBO registrarBO ?nomeCidadao ?numeroRG))
   (if (?numeroBO <> nil) then
        (assert(cidadao_registrou_BO ?nomeCidadao ?numeroRG ?numeroBO))
    )
)
```

5.4.5 A Inclusão de Fatos associados à Razão

O próximo passo consiste na inclusão na memória de trabalho do fato ou fatos para disparar a execução das primeiras regras de negócio. Estes fatos são aqueles associados à razão que embasa a realização do serviço. Por exemplo, a *"perda do RG"* é uma das razões

que embasam a realização do serviço *"Solicitar a Segunda Via do RG"* e como tal, um fato indicando que "cidadão perdeu o RG" deverá ser gerado na memória de trabalho. Por exemplo, para incluir o fato de que o "cidadão José Silva perdeu seu RG de número 123" executa-se um comando Jess, a partir do programa em Java, da seguinte forma:

bre.executeCommand("(assert (cidadaoPerdeuRG José Silva 123))");.

5.4.6 A Execução da Máquina de Regras Jess

Feito isto, o Executor executará a máquina de regras considerando o conteúdo da memória de trabalho que neste instante tem todas as regras e fatos necessários para disparar a execução de regras. A execução das regras de negócio, por sua vez, gerarão novos fatos, que vão disparar a execução de outras regras. Algumas destas regras terão como ação a chamada de serviços Web, que individualmente realizarão parte do serviço e no conjunto realizarão o serviço desejado pelo cidadão. Por exemplo, na regra de cancelamento de RG mostrada na Tabela 5-5, considerando-se que o "cidadão **não** registrou o BO", pelo mecanismo de encadeamento regressivo de Jess a outra regra que trata do registro de BO será disparada e, se a ação realizada pelo correspondente serviço Web for concluída com sucesso, ocorrerá a geração do novo fato "cidadao_registrou_BO". Ato contínuo, a regra anterior, de cancelamento de RG será agora disparada e a ação de cancelamento de RG realizada pelo respectivo serviço Web possibilitará a geração do novo fato "RG_foi_cancelado".

A idéia é que, com as regras de negócio associadas à necessidade de realização do serviço e pelo menos um fato associado à razão pela qual o serviço está sendo solicitado, o serviço possa ser iniciado. Este fato é que dará início à execução das regras de negócio que dispararão ações que poderão invocar a execução de Serviços Web que por sua vez poderão gerar novos fatos. Estes novos fatos darão início à execução de novas regras de negócio e assim, sucessivamente, novos Serviços Web serão executados, novos fatos serão gerados, até que o desejo do cidadão seja realizado, ou seja, o serviço seja concluído.

5.5 Considerações Finais do Capítulo 5

Este Capítulo apresentou a prova de conceito realizada para experimentar a exequibilidade das idéias e conceitos propostos nesta tese e forneceu uma visão geral das plataformas de desenvolvimento de regras e de execução de serviços.

A forma como foi realizada a prova de conceito mostrou que a abordagem proposta é promissora e exequível em termos de implementação das ideías e conceitos apresentados. A partir das condições de contorno e das decisões sobre como os aspectos da proposta foram considerados exercitou-se alguns aspectos de uso do IDE na criação e manutenção de termos, predicados e regras de negócio da comunidade de identificação Civil e alguns aspectos das etapas de preparação do serviço e de execução do serviço.

No Anexo C encontra-se um resumo desta visão geral do ponto de vista operacional do editor de regras, do preparador de serviços e do executor de serviços.

No próximo Capítulo serão tratadas as contribuições do trabalho e das necessidades de pesquisas futuras.

Capítulo 6

Conclusão

A necessidade de refletir rapidamente nos sistemas computacionais as mudanças nas regras que definem o comportamento das organizações colocam os engenheiros de software frente ao desafio de buscar uma solução que facilite a realização dessas mudanças. Notam-se esforços convergentes para enfrentar esse desafio, em diversas áreas de pesquisa, tais como em regras de negócio, serviços no contexto de SOA, mecanismos de composição de serviços, mecanismos de descoberta de serviços e ontologia, além de esforços na definição de modelos e metamodelos relacionados a estas áreas. As pesquisas nestas áreas têm procurado prover respostas adequadas, mas encontram-se ainda num nível de maturidade distante de soluções gerais, principalmente em termos de performance e segurança.

Este trabalho partiu da premissa de que o uso da tecnologia de regras de negócio combinadas com as tecnologias emergentes derivadas das áreas de pesquisas citadas provêem uma alternativa de solução muito mais flexível e promissora do que as abordagens tradicionais de desenvolvimento e execução de aplicações. A principal característica da tecnologia de regras de negócio é possibilitar que o conhecimento do negócio seja formalmente representado como regras que devem ser mantidas em uma base de conhecimento de forma que as regras de negócio e o restante da funcionalidade da aplicação possam ser atualizados de maneira independente. Ao externalizar os conhecimentos do negócio que estão embutidos nos procedimentos da aplicação as regras de negócio fornecem uma possibilidade de solução para o desenvolvimento de aplicações que precisam ser modificadas rapidamente. Porém, os mecanismos de regras existentes ainda não conseguem capturar as regras na terminologia de negócio e transformá-las em código executável sem a ajuda de engenheiros de software.

Este Capítulo trata das contribuições deste trabalho e das necessidades de pesquisas futuras. Na Seção 6.1 serão revistas as contribuições do trabalho citadas na Introdução,

acrescidas de alguns comentários que ilustram as propostas de soluções para os problemas. A seção 6.2 apresenta uma discussão sobre tópicos que merecem pesquisas futuras para aprofundar o debate ou derivar novas idéias.

6.1 Contribuição

Este trabalho teve como objetivo geral a proposição de um modelo flexível para desenvolvimento e execução de serviços (no sentido lato), baseado em regras de negócio, que consegue refletir rapidamente nos sistemas computacionais as mudanças demandadas no dia-a-dia dos negócios das empresas e dos governos.

Assim, as principais contribuições estão listadas a seguir:

- *Externalização de regras de negócio da lógica das aplicações para facilitar a manutenção de sistemas computacionais.*

Na grande maioria dos sistemas computacionais existentes as regras de negócio estão espalhadas na documentação e no código executável e é necessário separá-las para facilitar a manutenção, mas deixá-las preparadas para serem combinadas com outros processos de negócio, que podem incluir regras de negócio, porém são porções da lógica mais estáveis que as representadas pelas regras de negócio. A separação e identificação das porções dinâmicas e porções estáticas da lógica da aplicação possibilita externalizar e expressar as porções dinâmicas como regras de negócio e deixá-las independentes das porções estáticas que podem ser expressas como componentes de serviço. As regras de negócio, através de sua externalização, propiciam a independência necessária, para que as mudanças em porções altamente dinâmicas não causem efeitos colaterais em porções mais estáveis da lógica de negócio. Isto contribui para facilitar a manutenção e reduzir o tempo de manutenção de sistemas.

O modelo apóia-se em componentes de serviço específicos publicados em mecanismos universais de registro, tais como UDDI. Estes componentes de serviço implementam as lógicas e procedimentos específicos de negócio que são menos suscetíveis a mudanças das regras. Portanto, faz-se a combinação da execução de regras de negócio que contemplam mais os aspectos de controle e relacionamentos entre regras, que são dinâmicos e mais

suscetíveis a mudanças, com a execução de componentes de serviço que implementam lógicas específicas de negócio, que são menos suscetíveis a mudanças. Esta abordagem leva em consideração a existência de muitas aplicações já desenvolvidas (legadas), com porções bastante estáveis e, normalmente, de difícil manutenção e atualização. Assim, parte-se do princípio de que é possível o aproveitamento destas aplicações legadas transformando-as em serviços para facilitar o processo de integração na fase de projeto e de execução das regras de negócio.

- *Proposição de método de desenvolvimento que aproveita as vantagens dos mecanismos de regras de negócio e dos mecanismos de composição de serviços.*

Como uma extensão da contribuição anterior, da mesma forma que externaliza-se as regras de negócio de qualquer aplicação (existente, legada ou a construir) é possível externalizar as regras de negócio da lógica de composição, controlada pelos mecanismos de composição de processos, tais como BPEL. Esta abordagem, embora introduza uma sobrecarga com a inclusão de novos elementos, como novos programas-cliente e novas interfaces para a composição, tem a vantagem de a composição já estar na forma de serviço, o que facilita a integração com as regras de negócio. Assim, aproveita-se as vantagens dos mecanismos de regras, no que se refere à amigabilidade e externalização das regras e as vantagens dos mecanismos de composição de processos, no que se refere à padronização, facilidade de integração e independência de tecnologias.

- *Proposição de um método de transformação automática de regras de negócio descritas na terminologia das pessoas de negócio para regras executáveis.*

Inclui a proposição de modelos para formalizar a descrição de regras de negócio na terminologia das pessoas de negócio. Em qualquer comunidade de negócios existem termos e expressões que carregam uma semântica toda própria e são usados para representar objetos e fatos que são corriqueiros na comunidade. A descrição de regras de negócio numa linguagem de fácil assimilação para os analistas de negócio, que usa esses termos e expressões, facilita a manutenção na medida em que a mudança pode ser feita diretamente pelo analista de negócio e não pelo engenheiro de software. A formalização das regras de negócio facilita o processo de transformação automática para código executável ou para linguagens compreensíveis para engenheiros ou para mecanismos de regras proprietários.

- *Proposição de modelo para facilitar o intercâmbio de regras entre ferramentas heterogêneas.*

Embora a grande maioria dos metamodelos ainda não esteja completamente especificada, alguns deles ainda em fase de RFP, o modelo de desenvolvimento já considera ou deixa abertura para o uso de metamodelos de regras de negócio, tais como SBVR, PRR e SWRL. Esta consideração é muito importante, uma vez que estes metamodelos, por serem independentes de computação ou plataforma, facilitam o intercâmbio de modelos de regras entre ferramentas tecnologicamente heterogêneas, considerando a alta distributividade e complexidade das conexões entre as aplicações Web da atualidade.

Para verificar estas contribuições teóricas foi elaborada uma arquitetura de desenvolvimento e execução de serviços, baseada em regras de negócio e foi desenvolvida uma prova de conceito enfatizando os aspectos considerados fundamentais:

- A proposição de uma plataforma de desenvolvimento de regras de negócio, composta por um IDE de desenvolvimento de regras e por repositórios de vocabulários, regras e ontologias. O IDE proposto faz as seguintes contribuições inéditas em relação a outras iniciativas semelhantes em regras de negócio: (i) o ambiente dá suporte aos analistas de negócio na definição de regras, usando uma linguagem bastante familiar a eles, ou seja, usando os elementos e artefatos com os quais eles realizam seus negócios; (ii) o IDE, com o uso de modelos, consegue capturar as regras na terminologia das pessoas de negócio e expressá-las em modelos independentes de computação (CIM); (iii) ao definir os termos e predicados típicos que lhe são familiares, o analista de negócio começa a compor o vocabulário de sua comunidade, onde os termos e predicados serão a base para a definição dos fatos; (iv) o processo de definição de regras de negócio concretiza a individualização e externalização de porções dinâmicas da lógica da aplicação como regras de negócio e as condições e ações das regras são instanciadas a partir dos fatos definidos para a comunidade; (v) a possibilidade de fazer a ligação antecipada de regras de negócio com componentes de serviço relacionados, no instante de criação da regra, exclui a necessidade de intervenção do engenheiro de software para fazer a conexão; a

formalização de regras de negócio é baseada nas propostas do emergente metamodelo SBVR [19].

- A proposição de uma plataforma de execução orientada a serviços (no sentido lato), baseado em regras de negócio composto por um módulo preparador de execução e por um executor de regras de negócio. O modelo de execução proposto aproveita uma parte básica que é comum a muitas das propostas de arquiteturas de execução de regras de negócio [20]. O que não está presente na maioria das arquiteturas é a idéia em que se fundamenta a configuração e execução orientada a serviços. A plataforma de execução orientada a serviços também apresenta características inéditas em relação a outras iniciativas semelhantes: (i) a orientação a serviços refere-se ao fato de que a necessidade de realização de um serviço (no nível do negócio) é o evento que ativa a execução da plataforma; (ii) a principal característica que distingue o módulo preparador de execução é que ele faz uso de regras de negócio para orientar a descoberta ou necessidade de realização de serviços fornecendo uma interação eficiente e bastante amigável ao usuário; (iii) o módulo preparador evita desperdício de recursos computacionais ao fazer verificações de disponibilidade de serviços e de consistência de parâmetros para os serviços antes de se iniciar a execução, valendo-se da estrutura ontológica do serviço que será executado; (iv) a principal característica que distingue o módulo executor de serviços é que ele traduz em tempo de execução, as regras de negócio formalizadas em linguagem natural (modelos CIM) para modelos computacionais independentes de plataforma (PIM) ou para modelos específicos de plataforma (PSM), ou seja em regras computacionalmente executáveis; (v) o módulo executor faz a geração de programas-cliente para os componentes de serviço em tempo de execução; (vi) instancia e controla a execução de um mecanismo de regras, que invoca os componentes de serviço; (vii) ao traduzir as regras a cada necessidade de realização de serviço melhora a dinâmica da execução e facilita a manutenção na medida em que as regras de negócio podem ser atualizadas até instantes antes da execução; (viii) a geração de programas-cliente, a partir de parâmetros obtidos dos arquivos descritores e do Perfil de Serviço, para invocar operações nos componentes de serviço mostrou-se bastante promissora e simples se comparada com o grande número

de componentes auxiliares gerados por grande parte das plataformas e ambientes integrados existentes.

Resumidamente, os conceitos apresentados, as idéias e as plataformas propostas contribuem para solucionar os problemas delineados ao proverem as seguintes facilidades:

- Posicionar um método de desenvolvimento de vocabulários e regras de negócio descritos em linguagem de fácil assimilação por pessoas de negócio.
- Isolar as regras de negócio das porções mais estáveis das aplicações, flexibilizando a manutenção e a construção de sistemas computacionais.
- Posicionar um método de edição de regras de negócio que incluem ligações para componentes de serviço em tempo de modelagem.
- Posicionar um método de preparação e execução de serviços (no nível de negócios) com apoio de máquinas de execução de regras de negócio e de mecanismos para a descoberta e invocação de serviços.
- Posicionar uma plataforma de execução orientada a serviços. A orientação a serviços refere-se ao fato de que a necessidade de realização de um serviço é o evento que ativa a plataforma.
- Apoio de repositórios de vocabulários, regras de negócio e ontologias de serviços. Estes repositórios permitem que os recursos possam ser compartilhados e usados concorrentemente pelo ambiente de preparação e execução de serviços e pelo ambiente de desenvolvimento.

6.2 Pesquisas Futuras

Os resultados indicam que os conceitos, as idéias e as plataformas propostas sobre o modelo de desenvolvimento e execução de regras de negócio são promissoras. Além de tecnologias em regras de negócio, serviços (SOA), repositórios e ontologias, acredita-se que a solução completa para os problemas mencionados inclui o estudo e integração de outras tecnologias, tais como, transformação de modelos, geração automática de programas e compilação.

A seguir estão listados os tópicos que merecem ser focalizados em pesquisas futuras:

- Experimentação de múltiplas tecnologias, por exemplo, baseadas em repositórios de vocabulários, thesauris e ontologias combinadas com diálogos orientados por regras de negócio, com o objetivo de criar um modelo de delimitação do escopo de serviço, nos aspectos tratados pelo módulo Preparador.
- Consolidação de algoritmos de transformação de modelos para transformar regras de negócio aderente ao metamodelo SBVR para modelos independentes de plataforma (PIM) ou para modelos específicos de máquina de regras ou plataforma (PSM). A implementação destes métodos pode ser feita, por exemplo, usando-se tecnologias de transformação de esquemas ou de compilação.
- Prospecção de alternativas para a geração de código de programas-cliente (*proxies)* de serviços, baseado em arquivos descritores e Perfis de Serviço.
- Inclusão de mecanismo no IDE para contemplar modelagem de composição de processos usando, por exemplo linguagens como BPMN e que faça a transformação para linguagens executáveis como WS-BPEL.
- Consolidação de um método para realizar a descoberta automática de serviços baseada na declaração de fatos e condições, durante a criação de regras de negócio. Esta descoberta é muito útil na formalização de ontologia de serviços para estruturar as relações entre os serviços.
- Inclusão no Editor de uma outra alternativa para externalizar regras de negócio, por exemplo, usando conceitos de AOP, de maneira semelhante à proposta em composição híbrida [34].

- Definição e implementação de repositórios, aderentes ao metamodelo MOF, com linguagem de consulta e manipulação padronizada.
- Realizar a transformação de regras para a linguagem SWRL antecipando o surgimento de mecanismos de regras que executem esta linguagem padrão.
- Considerando a redução do domínio por comunidade de negócio estudar e propor uma maneira de fazer ligação tardia, ou seja, em tempo de execução, entre as regras de negócio e os componentes de serviço.
- Incluir mecanismos no IDE de Regras para analisar e aproveitar dados do Histórico de Execução.

Referências

[1] S. Hildreth. Rounding Up Business Rules. ComputerWorld. Julho/2005.

[2] J. Keping. A Business Rule Explanation System for Web Services. Dissertação de mestrado na University of New Brunswick. Fredericton, New Brunswick, Canada. NRC 46527. Maio/2003.

[3] Cuecent. Standard based Framework - Cuecent BPMS. http://www.bahwancybertek.com/cuecentBpms.html. Acesso em Outubro/2006.

[4] P. L. Alenquer. Regras de Negócio para Análise em Ambientes OLAP. Dissertação de Mestrado em Informática, UFRJ. Rio de Janeiro. 2002.

[5] L. L. Briggs. New Business Rules to Live By. Application Development Trends. http://www.adtmag.com/article.aspx?id=17961. Acesso em Outubro/2006.

[6] B. Sotomayor. A short introduction to Web Services. Department of Computer Science at the University of Chicago. http://gdp.globus.org/gt4-tutorial/multiplehtml/ch01s02.html. Acesso em Outubro/2006.

[7] Fair Isaac. Blaze Advisor. http://www.fairisaac.com/Fairisaac/Solutions/Enterprise+Decision+Management/Business+rules/Blaze+Advisor/Blaze+Advisor.htm Acesso em Outubro/2006.

[8] ILOG. Jrules. http://www.ilog.com. Acesso em Outubro/2005.

[9] E. Friedman-Hill. Jess in Action. Manning Publication Co. 2003.

[10] M. Barnes, D. Kelly. Play by the Rules. Byte (Special Report), Vol. 22, No. 6, pp. 98-102. 1997.

[11] D. K. Barry. Web Services and Service-Oriented Architectures: The Savvy Manager's Guide. Morgan Kaufmann. Abril/2003.

[12] H. He. What is Service-Oriented Architecture? O'Reilly WebServices.XML.com. Setembro/2003.

[13] W3C. Web Services Architecture. Working Group Note 11. Fevereiro/2004. http://www.w3.org/TR/ws-arch/. Acesso em Outubro/2005

[14] Q. H. Mahmoud. Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI). SDN - Sun Developer Network. Abril/2005. http://java.sun.com/developer/technicalArticles/WebServices/soa/index.html.

[15] Fair Isaac Corporation. Achieving Decision Consistency Across the SOA-based Enterprise: Using Business Rules Management Systems in an SOA. White Paper. Outubro/2005.

[16] C. Nagl, F. Rosenberg, S. Dustdar. ViDRE – A Distributed Service-Oriented Business Rule Engine based on RuleML. Abril/2006. http://www.infosys.tuwien.ac.at/Staff/rosenberg/papers/TR/TUV-1841-2006-38.pdf

[17] K. Geminiuc. A Services-Oriented Approach to Business Rules Development - SOA Best Practices: The BPEL Cookbook. Oracle Technology Network. http://www.oracle.com/technology/pub/articles/bpel_cookbook/geminiuc.html. Acesso em outubro/2006).

[18] S. Hildreth. Rounding Up Business Rules. ComputerWorld Software. IDG. Maio/2005.http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=101844&pageNumber=3.

[19] OMG. SBVR - Semantics of Business Vocabulary and Business Rules - Adopted Specification. Março/2006. http://www.omg.org/docs/dtc/06-03-02.pdf.

[20] M. Chisholm. How to Build a Business Rules Engine. Elsevier. Morgan Kauhfmann Publishers. 2004.

[21] OMG - The Object Management Group http://www.omg.org/.

[22] W3C - The World Wide Web Consortium. http://www.w3.org/

[23] R. G. Ross. What about IT projects? - Principles of Business Rule Approach. Addison-Wesley. 2003.

[24] Y. Malhotra. Knowledge Management for the New World of Business. 1998. http://www.brint.com/km/whatis.htm . Acesso em Outubro/2006.

[25] B. V. Halle. Business Rules Applied. Wiley, Primeira Edição. 2001.

[26] M. Afshar, B. Nainani. Building Flexible Business Processes Using BPEL and Rules. Don't embed business policies in your business processes: Automate them with a business rules engine!. Dezembro/2005.

[27] C. J. Date. What Not How. Addison-Wesley Professional. Abril/2000.

[28] InfoSapient. The Representation and Execution of Business Operation Rules. White Paper. 2001. http://info-sapient.sourceforge.net/White_Paper/BusinessProcessRules.pdf.

[29] R. Burlton. From Strategic Intent to Requirements Definition. Business Rules Forum. 2004.

[30] J. Zachman. Business Rules and the Framework, Semantic Arts. http://semarts.com.decisivenet.com/DesktopModules/ViewArticle.aspx?ArticleID=723&mid=3476, 2003.

[31] J. Sinur, WITH-19-9972, Notices of Research, 2003.

[32] OASIS. Web Services Business Process Execution Language Version 2.0. Maio/2006. http://www.oasis-open.org/apps/group_public/download.php/18714/wsbpel-specification-draft-May17.htm Acesso em Outubro/2006.

[33] D. Hollingsworth. The Workflow Reference Model 10 Years On. WfMC Technical Committee. http://www.wfmc.org/standards/docs/Ref_Model_10_years_on_Hollingsworth.pdf Acesso em Outubro/2006.

[34] A. Charfi, M. Mezini. Hybrid Web Service Composition: Business Processes Meet Business Rules. ICSOC'04, November 15–19, 2004, New York. http://delivery.acm.org/10.1145/1040000/1035173/p30-charfi.pdf?key1=1035173&key2=6547341511&coll=ACM&dl=ACM&CFID=15151515&CFTOKEN=61846188 Acesso em Outubro/2006.

[35] N.G.T. DeSilva., R.I. Pushpaka, K.S. Kumar, H.M.R.S. Dayananda. Business Rules Engine for Java. White paper - Department of Computer Science and Engineering University of Moratuwa. 2003. http://www.cse.mrt.ac.lk/projects/p3-2003-2nd/papers/paper-grp-06-Mrules_White%20paper.doc Acesso em Outubro/2006.

[36] T. Elrad, M. Aksit, S. Clarke, R. E. Filman. Introduction to Aspect-Oriented Software Development. Addison Wesley Professional..Outubro/2004.

[37] M. D'hondt. Hybrid Aspects for integrating Rule-based Knowledge and Object-Oriented Functionality. Tese de doutorado. Vrije Universiteit Brussel, Maio/2004.

[38] A. Charfi, M. Mezini. Aspect Oriented Web Service Composition. Proceedings of the European Conference on Web Services. ECOWS 2004. LNCS 3250.

[39] E. Friedman-Hill. Jess in Action. Manning. 2003.

[40] B. Orriens, J. Yang. Specification and Management of Policies in Service Oriented Business Collaboration. Tilburg University, Netherlands, Macquarie University,Sydney. infolab.uvt.nl/pub/orrinsb-2005-109.pdf. Acesso em Outubro/2006.

[41] RuleML. The Rule Markup Initiative. http://www.ruleml.org/. Acesso em Outubro/2006.

[42] Bloor Research. ILOG Jrules 6.0. White Paper. http://www.ilog.com/products/jrules/whitepapers/. 2006. Acesso em Outubro/2006.

[43] D. Jobst, R. V. Ammon, B. Gebauer. Business Rules Engines Within Enterprise Platforms. ILOG Jrules 6. Abril/2005.

[44] Java community Porcess. JSRs: Java Specification Requests, JSR-94 Java™ Rule Engine API. http://jcp.org/en/jsr/detail?id=94. Acesso em Outubro/2006.

[45] A. Houaiss. Dicionário Eletrônico Houaiss da Lingua Portuguesa. Editora Objetiva Ltda. Dezembro/2001.

[46] E. P. Flores. O conceito de regra na linguagem cotidiana e na Análise Experimental do Comportamento. Universidade de Brasília. Estudos de Psicologia, 9(2), 279-283. 2004.

[47] T. H. Davenport, L. Prusak. Conhecimento Empresarial: Como as organizações gerenciam seu capital intelectual. Rio de Janeiro. Campus, 1998.

[48] BRG. Defining Business Rules ~ What Are They Really?. BRG-Business Rules Group é o sucessor de GUIDE Business Rules Project. Julho/2000. http://rewerse.net/downloads/BRG-whatisBR_3ed.pdf. Acesso em Outubro/2006.

[49] IBM. Initial Submission to BSBR - Business Semantics of Business Rules. Janeiro/2004.

[50] G. Wagner. How to design a general rule markup language? Workshop XML Technologien fuer das SemanticWeb (XSW), Berlin, Junho/2002.

[51] D. C. Hay. Modeling Business Rules: What Data Models Do. Essential Strategies. The Data Administration Newsletter (TDAN.com). Janeiro/2004.

[52] R. G. Ross. Serving Up Knowledge. Principles of Business Rule Approach.Addison-Wesley, 2003

[53] B. K. O'Neil. Business Metadata: How to write definitions. The Data Administration Newsletter (TDAN.com). Abril/2005. http://www.tdan.com/i032fe01.htm Acesso em Outubro/2006.

[54] OASIS - Organization for the Advancement of Structured Information Standards. http://www.oasis-open.org/who/. Acesso em Outubro/2006.

[55] OASIS. UDDI - Universal Description, Discovery, and Integration Version 3. Fevereiro/2005. http://xml.coverpages.org/ni2005-02-02-a.html. Acesso em Outubro/2006.

[56] DAML - The DARPA Agent Markup Language. http://www.daml.org/. Acesso em Outubro/2006.

[57] DAML. OWL-S 1.2 Pre-Release. Março/2006. http://www.daml.org/services/owl-s/. Acesso em Outubro/2006.

[58] K. Birman. Like It or Not, Web Services Are Distributed Objects. Communications of the ACM. Vol. 47, No. 12, 60-62. Dezembro/2004.

[59] W. Vogels, Web Services Are Not Distributed Objects. IEEE Internet Computing. 7, 6, 59–66. Novembro–Dezembro/2003.

[60] Wikipedia - The free encyclopedia. Wikipedia:Ontology. http://en.wikipedia.org/wiki/Ontology. Acesso em Outubro/2006.

[61] A. Barr, E.A. Feigenbaum. *The Handbook of Artificial Intelligence* , vol. 1, William Kaufmann, 1981.

[62] D.N. Chorafas. *Agent Technology Handbook*. McGraw-Hill, New York, 1998.

[63] S. Russel, P. Norvig. Artificial Intelligence: A Modern Approach. Englewood Cliffs, NJ: Prentice-Hall. 1995.

[64] E. Rich, K. Knight. Inteligência Artificial. São Paulo, SP: Makron Books do Brasil. 1994

[65] T. Lindgren. Methods for Rule Conflict Resolution. Machine Learning: ECML 2004. Lecture Notes in Computer Science. 2004.

[66] BPMI - Business Process Management Initiative. http://www.bpmi.org/. Acesso em Outubro/2006.

[67] W3C. Extensible Markup Language (XML). http://www.w3.org/XML/ Acesso em Outubro/2006.

[68] OMG. **The Unified Modeling Language. http://www.omg.org/UML/.** Acesso em Outubro/2006.

[69] W3C. OWL Web Ontology Language Overview. W3C Recommendation. Fevereiro/2004. http://www.w3.org/TR/owl-features/. Acesso em Outubro/2006.

[70] W3C. RDF Primer. W3C Recommendation 10. Fevereiro/2004. http://www.w3.org/TR/rdf-primer/ . Acesso em Outubro/2006.

[71] W3C. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10. Fevereiro/2004. http://www.w3.org/TR/rdf-schema/. Acesso em Outubro/2006.

[72] D. Hirtle, H. Boley, B. Grosof, M. Kifer, M. Sintek, S. Tabet, G. Wagner. Schema Specification of RuleML 0.91. Agosto/2006. http://www.ruleml.org/0.91/. Acesso em Outubro/2006.

[73] Wikipedia. SQL - Structured Query Language. http://en.wikipedia.org/wiki/SQL. Acesso em Outubro/2006.

[74] OMG. Object Constraint Language Version 2.0, formal/06-05-01. Maio/2006.

[75] OASIS – Cover Pages. Xlang. Technology Reports. Junho/2001. http://xml.coverpages.org/xlang.html. Acesso em Outubro/2006.

[76] W3C. XQuery 1.0: An XML Query Language, W3C Proposed Recommendation. Novembro/2006. http://www.w3.org/TR/2006/PR-xquery-20061121/. Acesso em Novembro/2006.

[77] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl. RQL: A Declarative Query Language for RDF. WWW2002, Honolulu, USA, Maio/2002.

[78] W3C Member Submission. OWL-S: Semantic Markup for Web Services, Novembro/2004. http://www.w3.org/Submission/OWL-S/. Acesso em Outubro/2006.

[79] Wikipédia. Prolog. http://pt.wikipedia.org/wiki/Prolog. Acesso em Outubro/2006.

[80] Mandarax, The Mandarax Project. http://mandarax.sourceforge.net/. Acesso em Maio/2006).

[81] T. Kawamura, J. A. De Blasio, T. Hasegawa, M. Paolucci, K. Sycara. Preliminary Report of Public Experiment of Semantic Service Matchmaker with UDDI Business Registry. Research and Development Center, Toshiba Corp, The Robotics Institute, Carnegie Mellon University.

[82] W3C. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission. Maio/2004. http://www.w3.org/Submission/SWRL/. Acesso em Outubro/2006.

[83] S. Tabet, G. Wagner, S. Spreeuwenberg, P. Vincent, G. Jacques, C. de Sainte Marie, J. Pellant, J. Frank, J. Durand. OMG Production Rule Representation - Context and Current Status. W3C Workshop on Rule Languages for Interoperability. Março/2005.

[84] OMG. Production Rule Representation Request For Proposal OMG Document: br/2003-09-03. Setembro/2003. http://www.omg.org/docs/br/03-09-03.pdf. Acesso em Outubro/2006.

[85] OMG. RIF Use Cases and Requirements. W3C Working Draft. Julho/2006. http://www.w3.org/TR/rif-ucr/. Acesso em Outubro/2006.

[86] [MOF2]

[87] D. Selman, J. Majoor. The Java Community and Rule Engine Standards. 2004. http://64.233.161.104/search?q=cache:ty23HlpFj64J:www.w3.org/2004/12/rules-ws/paper/107/+JSR+94+Blaze+ilog+Aion&hl=pt-BR&gl=br&ct=clnk&cd=1. Acesso em Outubro/2006.

[88] Drools. Implementation of Forgy's Rete algorithm, http://drools.org/. Acesso em Maio/2006.

[89] Computer Associates. CleverPath Aion r10. 2005. http://www3.ca.com/Press/PressRelease.aspx?CID=67053. Acesso em Outubro/2006.

[90] Haley Technology. Haley Systems Advances Business Rule Implementation Standards for JSR-94. Press Coverage/Press Releases. Abril/2005.

http://www.haley.com/1972524733701122/newsevents/PressRelease_JSR94_Release_Final_200604.html. Acesso em Outubro/2006.

[91] Pegasystems. PegaRULES. http://www.pegasystems.com/. Acesso em Outubro/2006.

[92] OMG. BSBR - Business Semantics of Business Rules RFP - br/2003-06-03. Julho/2003.

[93] OMG. Meta Object Facility (MOF) Core Specification. Version 2.0. Janeiro/2006. http://www.omg.org/docs/formal/06-01-01.pdf. Acesso em Outubro/2006.

[94] OMG. MOF 2.0/XMI Mapping Specification, v2.1. Setembro/2005. http://www.omg.org/docs/formal/05-09-01.pdf. Acesso em Outubro/2006.

[95] [CIM]

[96] OMG. MDA Guide Version 1.0.1. Junho/2003. http://www.omg.org/docs/omg/03-06-01.pdf . Acesso em Outubro/2006.

[97] BRG. The Business Rules Manifesto. 2003. http://www.businessrulesgroup.org/brmanifesto.htm. Accesso em Maio/2006.

[98] BRG. Business Rules Group. http://www.businessrulesgroup.org/home-brg.shtml. Acesso em Outubro/2006.

[99] OMG. SBVR Annex M - Mappings and Relationships to Other Initiatives- Mapping to Other Standards and Metamodels. 2006.

[100] D. Gisolfi. Web services architect, Part 3: Is Web services the reincarnation of CORBA?. Julho/2001. http://www-128.ibm.com/developerworks/webservices/library/ws-arc3/ . Acesso em Outubro/2006.

[101] WfMC.The Workflow Management Coalition. http://www.wfmc.org/. Acesso em Outubro/2006.

[102] XPDL XML Process Definition Language. Version 2.00. Document Number WFMC-TC-1025. Document Status – Final. Outubro/2005

[103] R. Bortolini. Padronizando Processos: BPMN, BPML, XPDL e BPEL. Janeiro/2006. http://www.cryo.com.br/Site/Page/article.aspx?CC=54e3a8f3-d7ef-4941-b039-f982b8f5705c. Acesso em Outubro/2006.

[104] OASIS. Business Process Modeling Language Overview. XML Cover Pages. Agosto/2003. http://xml.coverpages.org/bpml.html. Acesso em Outubro/2006.

[105] [WSDL]

[106] OMG. BPMN - Business Process Modeling Notation Specification. OMG Final Adopted Specification. dtc/06-02-01. Fevereiro/2006.

http://www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf . Acesso em Outubro/2006.

[107] OMG. UML Unified Modeling Language: Superstructure. version 2.0. Agosto/2005. http://www.omg.org/docs/formal/05-07-04.pdf. Acesso em Outubro/2006.

[108] OMG. UML Profile for Enterprise Distributed Object Computing. Enterprise Collaboration Architecture (ECA) Specification. Fevereiro/2004. http://www.omg.org/docs/formal/04-02-01.pdf. Acesso em Outubro/2006.

[109] Federal Information Processing Standards Publications (FIPS PUBS). Integration Definition for Function Modeling (IDEF0). Dezembro/1993. http://www.itl.nist.gov/fipspubs/idef02.doc. Acesso em Outubro/2006.

[110] UN/CEFACT, OASIS. ebXML BPSS - ebXML Business Process Specification Schema. Version 1.0.1. Maio/2001. http://www.ebxml.org/specs/ebBPSS.pdf. Acesso em Outubro/2006.

[111] IBM. WebSphere Business Modeler - ADF (Activity Decision Flow). http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.btools.help.modeler.doc/doc/reference/migration/modelreplacements.html. Acesso em Outubro/2006.

[112] W3C. Web Services Choreography Description Language Version 1.0. W3C Candidate Recommendation Novembro/2005. http://www.w3.org/TR/ws-cdl-10/. Acesso em Outubro/2006.

[113] A. Kamada, M. Mendes. Business Rule Engine Applied to eGovernment Services Integration, I3E / IFIP 2005, Poznan, Poland, October/2005.

[114] A. Kamada, M. Mendes. Rule based flexible eGovernment applications, International Conference on E-Government, Ottawa, Canadá, 2005.

[115] R. G. Ross. Sentence Patterns for Rule Statements. Principles of Business Rule Approach, Addison-Wesley, 2003.

[116] [UDDI].

[117] DSTC Pty Ltd. dMOF 1.1 User Guide. http://www.dstc.com/Downloads/CORBA/MOF/dMOF1 1.UserGuide.pdf. 2002. Acesso em Outubro/2006.

[118] J. Butler, R. Hubbly, W. Melo. An MOF-based repository for enterprise architecture models. IBM DevelopersWork. Março/2005. http://www-128.ibm.com/developerworks/rational/library/mar05/melo/index.html. Acesso em Outubro/2006.

[119] M. Matula. NetBeans Metadata Repository. NetBeans.org open source project. Março/2003. http://mdr.netbeans.org/MDR-whitepaper.pdf. Acesso em Outubro/2006.

[120] A. M. C. M. Figueiredo, A. Kamada, M. J. Mendes, M. A. Rodrigues, L. Damasceno. Using metamodells to promote Data Integration in e-Government Application Scenario. Digital Communities in a Networked Society: e-Commerce, e-Business and e-Government. Norweel: Kluwer Academic, 2004, p. 293-303.

[121] Software AG. Unlock data and functions in your legacy systems. CrossVision. 2006

[122] W3C. OWL Guide OWL Web Ontology Language Guide. W3C Recommendation. Fevereiro/2004. http://www.w3.org/TR/owl-guide/. Acesso em Outubro/2006.

[123] A. S. Tanenbaum. Sistema Operacionais Modernos. Prentice-Hall do Brasil. 1992.

[124] eGOIA - Electronic Government Innovation and Access. http://www.egoia.info/. Acesso em Outubro/2006.

[125] CMU Sphinx Group. The CMU Sphinx Group Open Source Speech Recognition Engines. http://cmusphinx.sourceforge.net/html/cmusphinx.php. Acesso em Outubro/2006.

[126] S. Carter, A. Hurst, J. Mankoff, J. Li. Dynamically Adapting GUIs to Diverse Input Devices. *ASSETS'06,* Outubro/2006, Portland, USA. http://www.cs.cmu.edu/~assist/publications/06Carterassets-final-updated.pdf Acesso em Outubro/2006.

[127] CMU – RoomLine. RoomLine – A Spoken Dialogue System for Conference Room Reservation in SCS. Carnegie Mellon University. http://www.cs.cmu.edu/~dbohus/RoomLine/sample1.html. Acesso em Outubro/2006.

[128] Nuance. Dragon NaturallySpeaking 9: Turn talk into type. http://www.nuance.com/talk/ . Acesso em Outubro/2006.

[129] IBM. IBM ViaVoice, http://www-306.ibm.com/software/voice/viavoice/. Acesso em Outubro/2006.

[130] M. Horridge, H. Knublauch, A. Rector, R. Stevens, C. Wroe. A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0. The University Of Manchester, Stanford University. Agosto/2004. http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf. Acesso em Outubro/2006.

[131] Jena 2, A Semantic Web Framework, http://www.hpl.hp.com/semweb/jena2.htm. Acesso em Maio/2006.

[132] eXist. Overview - Open Source Native XML Database. http://exist.sourceforge.net/ acesso em: Setembro/2006.

[133] Xindice. Apache Xindice http://xml.apache.org/xindice/index.pdf acesso em: Setembro/2006.

[134] P. Ford. Berkeley DB XML: An Embedded XML Database. Maio/2003. http://www.xml.com/pub/a/2003/05/07/bdb.html Acesso em: Setembro/2006.

Anexo A – The Business Rules Manifesto*

Article 1.	***Primary Requirements, Not Secondary***
1.1.	Rules are a first-class citizen of the requirements world.
1.2.	Rules are essential for, and a discrete part of, business models and technology models.
Article 2.	***Separate From Processes, Not Contained In Them***
2.1.	Rules are explicit constraints on behavior and/or provide support to behavior.
2.2.	Rules are not process and not procedure. They should not be contained in either of these.
2.3.	Rules apply *across* processes and procedures. There should be one cohesive body of rules, enforced consistently across all relevant areas of business activity.
Article 3.	***Deliberate Knowledge, Not A By-Product***
3.1.	Rules build on facts, and facts build on concepts as expressed by terms.
3.2.	Terms express business concepts; facts make assertions about these concepts; rules constrain and support these facts.
3.3.	Rules must be explicit. No rule is ever assumed about any concept or fact.
3.4.	Rules are basic to what the business knows about itself -- that is, to basic business knowledge.
3.5.	Rules need to be nurtured, protected, and managed.
Article 4.	***Declarative, Not Procedural***
4.1.	Rules should be expressed declaratively in natural-language sentences for the business audience.
4.2.	If something cannot be expressed, then it is not a rule.

4.3. A set of statements is declarative only if the set has no implicit sequencing.

4.4. Any statements of rules that require constructs other than terms and facts imply assumptions about a system implementation.

4.5. A rule is distinct from any enforcement defined for it. A rule and its enforcement are separate concerns.

4.6. Rules should be defined independently of responsibility for the *who*, *where*, *when*, or *how* of their enforcement.

4.7. Exceptions to rules are expressed by other rules.

Article 5. ***Well-Formed Expression, Not Ad Hoc***

5.1. Business rules should be expressed in such a way that they can be validated for correctness by business people.

5.2. Business rules should be expressed in such a way that they can be verified against each other for consistency.

5.3. Formal logics, such as predicate logic, are fundamental to well-formed expression of rules in business terms, as well as to the technologies that implement business rules.

Article 6. ***Rule-Based Architecture, Not Indirect Implementation***

6.1. A business rules application is intentionally built to accommodate continuous change in business rules. The platform on which the application runs should support such continuous change.

6.2. Executing rules directly -- for example in a rules engine -- is a better implementation strategy than transcribing the rules into some procedural form.

6.3. A business rule system must always be able to explain the reasoning by which it arrives at conclusions or takes action.

6.4. Rules are based on truth values. How a rule's truth value is determined or maintained is hidden from users.

6.5. The relationship between events and rules is generally many-to-many.

Article 7. ***Rule-Guided Processes, Not Exception-Based Programming***

7.1. Rules define the boundary between acceptable and unacceptable business activity.

	7.2.	Rules often require special or selective handling of detected violations. Such rule violation activity is activity like any other activity.
	7.3.	To ensure maximum consistency and reusability, the handling of unacceptable business activity should be separable from the handling of acceptable business activity.
Article 8.		***For the Sake of the Business, Not Technology***
	8.1.	Rules are about business practice and guidance; therefore, rules are motivated by business goals and objectives and are shaped by various influences.
	8.2.	Rules always cost the business something.
	8.3.	The cost of rule enforcement must be balanced against business risks, and against business opportunities that might otherwise be lost.
	8.4.	'More rules' is not better. Usually fewer 'good rules' is better.
	8.5.	An effective system can be based on a small number of rules. Additional, more discriminating rules can be subsequently added, so that over time the system becomes smarter.
Article 9.		***Of, By, and For Business People, Not IT People***
	9.1.	Rules should arise from knowledgeable business people.
	9.2.	Business people should have tools available to help them formulate, validate, and manage rules.
	9.3.	Business people should have tools available to help them verify business rules against each other for consistency.
Article 10.		***Managing Business Logic, Not Hardware/Software Platforms***
	10.1.	Business rules are a vital business asset.
	10.2.	In the long run, rules are more important to the business than hardware/software platforms.
	10.3.	Business rules should be organized and stored in such a way that they can be readily redeployed to new hardware/software platforms.

10.4. Rules, and the ability to change them effectively, are fundamental to improving business adaptability.

*Version 2.0, November 1, 2003. Edited by Ronald G. Ross.

Anexo B – Regras em Notação SBVR traduzidas para Jess

As regras de negócio a seguir compõem o conjunto de regras para o serviço de "Solicitação de Segunda Via de RG por perda", em notação aderente ao metamodelo SBVR.

Um cidadão **pode** solicitar Segunda Via do RG **se todas as seguintes situações ocorrerem:** • cidadão perdeu **o** RG • cidadão registrou BO • RG está cancelado • cidadão pagou **a** taxa de serviço **ou** cidadão tem isenção de taxa de serviço.
É necessário cidadão registrar **um** BO **se** cidadão perdeu **o** RG.
Se dados RG existem então RG está cadastrado.
É obrigatório RG ser cancelado **se todas as seguintes situações ocorrerem:** • cidadão perdeu **o** RG • cidadão registrou BO • RG está cadastrado.
É obrigatório cidadão pagar **a** taxa de serviço **se todas as seguintes situações ocorrerem**: • cidadão perdeu **o** RG • cidadão **não** tem isenção de taxa de serviço.
É obrigatório cidadão ter isenção de taxa de serviço **se pelo menos uma das seguintes situações ocorrerem:** • cidadão teve seu RG roubado • idade do cidadão é maior ou igual a 60 anos • cidadão apresentou Atestado de Pobreza

As regras de negócio a seguir compõem o conjunto de regras para o serviço de "Solicitação de Segunda Via de RG por perda", em comandos Jess.

Regras sobre a solicitação de Segunda Via de RG, em Jess.

```
(do-backward-chaining RE_esta_cancelado)
(do-backward-chaining cidadao_registrou_BO)

 (defrule rule_solicitarSegundaViaRG
"Um cidadão pode solicitar Segunda Via do RG se todas as seguintes situações ocorrerem:
• cidadão perdeu o RG
```

- cidadão registrou BO
- RG está cancelado.
- cidadão pagou **a** taxa de serviço **ou** cidadão tem isenção de taxa de serviço**"**

```
   (cidadao_perdeu_RG ?nomeCidadao ?numeroRG)
   (cidadão_registrou_BO ?nomeCidadao ?numeroBO)
   (RG_esta_cancelado ?numeroRG)
   (or (cidadao_pagou_taxa_de_servico ?nomeCidadao ?numeroDocumento)
       (cidadão_tem_isencao_de_taxa_de_servico ?nomeCidadao))
   =>
   (defclass sProxyRG  ProxyRG)
   (bind ?proxyRG (new ProxyRG))
   (definstance sProxyRG ?proxyRG static)
   (if (call ?proxyRG solicitarSegundaViaRG ?numeroRG) then
      (assert (cidadao_solicitou_segunda_via_RG ?nomeCidadao ?numeroRG))
   else
      (printout t "Erro em serviço Web 'RG.solicitarSegundaViaRG', na chamada via ProxyRG" crlf)
      (printout t "Erro - elementos envolvidos: " ?nomeCidadao ", " ?numeroRG ", " ?numeroBO crlf)
   )
)

(defrule rule_registrarBO
"É necessário cidadão registrar um BO se cidadão perdeu o RG."

   (cidadao_perdeu_RG ?nomeCidadao ?numeroRG)
   =>
   (defclass sProxyBO ProxyBO)
   (bind ?proxyBO (new ProxyBO))
   (definstance sProxyBO ?proxyBO static)
   (bind ?numeroBO (call ?proxyBO registrarBO ?nomeCidadao ?numeroRG))
   (if (<> ?numeroBO nil) then
      (assert(cidadao_registrou_BO ?nomeCidadao ?numeroRG ?numeroBO))
   else
      (printout t "Erro em serviço Web BO.registrarBO', na chamada via ProxyBO" crlf)
      (printout t "Erro - elementos envolvidos: " ?nomeCidadao ", " ?numeroRG ", " ?numeroBO crlf)
   )
)

(defrule rule_cancelarRG
"É obrigatório RG ser cancelado se todas as seguintes situações ocorrerem:
•  cidadão perdeu o RG
•  cidadão registrou BO
•  RG está cadastrado. "

   (cidadao_perdeu_RG ?nomeCidadao ?numeroRG)
   (cidadao_registrou_BO ?nomeCidadao ?numeroRG ?numeroBO)
   (RG_esta_cadastrado ?nomeCidadao ?numeroRG)
   =>
   (defclass sProxyRG  ProxyRG)
   (bind ?proxyRG (new ProxyRG))
   (definstance sProxyRG ?proxyRG static)
   (bind ?result (call ?proxyRG cancelarRG ?numeroRG))
   (if (= ?result TRUE) then
```

```
      (assert (RG_esta_cancelado ?numeroRG))
   else
      (printout t "Erro em serviço Web RG.cancelarRG', na chamada via ProxyRG" crlf)
      (printout t "Erro - elementos envolvidos: " ?nomeCidadao ", " ?numeroRG ", " ?numeroBO crlf)
   )
)

(defrule rule_pagarTaxaServico
"É obrigatório cidadão pagar a taxa de serviço se todas as seguintes situações ocorrerem:
• cidadão perdeu o RG
• cidadão não tem isenção de taxa de serviço."

   (cidadao_perdeu_RG ?nomeCidadao ?numeroRG)
   (not (cidadão_tem_isencao_de_taxa_de_servico ?nomeCidadao))
   =>
   (defclass sProxyTaxaServico  ProxyTaxaServico)
   (bind ?proxyTaxaServico (new ProxyTaxaServico))
   (definstance sProxyTaxaServico ?proxyTaxaServico static)
   (bind ?result (call ?proxyTaxaServico pagarTaxaServico ?numeroDocumento))
   (if (= ?result TRUE) then
      (assert (cidadao_pagou_taxa_de_servico ?nomeCidadao ?numeroDocumento))
   else
      (printout t "Erro em serviço Web TaxaServico.pagarTaxaServico', na chamada via ProxyTaxaServico" crlf)
      (printout t "Erro - elementos envolvidos: " ?nomeCidadao ", " ?numeroRG ", " ?numeroDocumento crlf)
   )
)

(defrule rule_isentarTaxaServico
"É obrigatório cidadão ter isenção de taxa de serviço se pelo menos uma das seguintes situações
ocorrerem:
• cidadão teve seu RG roubado
• idade do cidadão é maior que 60 anos
• cidadão apresentou Atestado de Pobreza ."

   (or (cidadao_teve_RG_roubado ?nomeCidadao ?numeroRG)
       (idade_cidadão_maior_que_60_anos ?nomeCidadao ?idadeCidadao&: (> ?idadeCidadao 60))
       (cidadao_apresentou_atestado_de_pobreza  ?nomeCidadao ?numeroAtestado))
   =>
   (defclass sProxyTaxaServico  ProxyTaxaServico)
   (bind ?proxyTaxaServico (new ProxyTaxaServico))
   (definstance sProxyTaxaServico ?proxyTaxaServico static)
   (bind ?result (call ?proxyTaxaServico isentarTaxaServico ?nomeCidadao ?idadeCidadao ?numeroAtestado))
   (if (= ?result TRUE) then
      (assert (cidadao_tem_isencao_de_taxa_de_servico ?nomeCidadao))
   else
      (printout t "Erro em serviço Web TaxaServico.isentarTaxaServico', na chamada via ProxyTaxaServico" crlf)
      (printout t "Erro - elementos envolvidos: " ?nomeCidadao ", " ?idadeCidadao ", " ?numeroAtestado crlf)
   )
)

(defrule find-RGEstaCancelado
"Regra que procura satisfazer pré-condição não satisfeita em outra regra. Disparada através do mecanismo
backward-chaining."
```

```
    (need-RGEstaCancelado ?numeroRG)

   ;Dica para transformar: se este padrão em LHS desta regra, a menos do prefixo "need_"
   ;(no caso, "RGEstaCancelado ?numeroRG") é asserção em RHS de outra regra (no caso
   ;"rule_cancelarRG"), então incluir a partir daqui todo o conteúdo desta outra regra.

   (cidadao_perdeu_RG ?nomeCidadao ?numeroRG)
   (cidadao_registrou_BO ?nomeCidadao ?numeroRG ?numeroBO)
   (RG_esta_cadastrado ?nomeCidadao ?numeroRG)
   =>
   (defclass sProxyRG  ProxyRG)
   (bind ?proxyRG (new ProxyRG))
   (definstance sProxyRG ?proxyRG static)
   (bind ?result (call ?proxyRG cancelarRG ?numeroRG))
   (if (= ?result TRUE) then
      (assert (RG_esta_cancelado ?numeroRG))
   else
      (printout t "Erro em serviço Web RG.cancelarRG', na chamada via ProxyRG" crlf)
      (printout t "Erro - elementos envolvidos: " ?nomeCidadao ", " ?numeroRG ", " ?numeroBO crlf)
   )
)

 (defrule find-cidadao_registrou_BO
"Regra que procura satisfazer pré-condição não satisfeita em outra regra. Disparada através do mecanismo
backward-chaining."

   (need-cidadao_registrou_BO ?nomeCidadao ?numeroRG ?)
   =>
   (defclass sProxyBO ProxyBO)
   (bind ?proxyBO (new ProxyBO))
   (definstance sProxyBO ?proxyBO static)
   (bind ?numeroBO (call ?proxyBO registrarBO ?nomeCidadao ?numeroRG))
   (if (<> ?numeroBO nil) then
      (assert(cidadao_registrou_BO ?nomeCidadao ?numeroRG ?numeroBO))
   else
      (printout t "Erro em serviço Web BO.registrarBO', na chamada via ProxyBO" crlf)
      (printout t "Erro - elementos envolvidos: " ?nomeCidadao ", " ?numeroRG ", " ?numeroBO crlf)
   )
)

 (defrule find-RGEstaCadastrado
"Regra que procura satisfazer pré-condição não satisfeita em outra regra. Disparada através do mecanismo
encadeamento regressivo."

  (need-RGEstaCadastrado ?numeroRG)

  ;Dica para transformar: se este padrão em LHS desta regra, a menos do prefixo "need_"
  ;(no caso, "RGEstaRegistrado ?numeroRG") é asserção em RHS de outra regra (no caso
  ;"rule_registrarRG"), então incluir a partir daqui todo o conteúdo desta outra regra.

  (not (RGEstaCadastrado ?numeroRG))
  =>
   =>
   (defclass sProxyRG  ProxyRG)
```

```
   (bind ?proxyRG (new ProxyRG))
   (definstance sProxyRG ?proxyRG static)
   (bind ?result (call ?proxyRG cadastrarRG ?numeroRG))
   (if (= ?result TRUE) then
      (assert (RG_esta_cadastrado ?numeroRG))
   else
      (printout t "Erro em serviço Web RG.cadastrarRG', na chamada via ProxyRG" crlf)
      (printout t "Erro - elementos envolvidos: " ?nomeCidadao ", " ?numeroRG crlf)
   )
)
```

Anexo C – Visão Geral de Uso das Plataformas

Este Anexo provê uma visão geral de uso das plataformas de desenvolvimento de regras e de execução de serviços. A Seção C.1 apresenta um roteiro de como desenvolver regras de negócio, fatos e termos associados a uma comunidade, usando o Editor de Regras. A Seção C.2 apresenta a sequência de ações que ocorrem quando um serviço é executado.

C.1 Desenvolvimento de Regras de Negócio

O IDE comporta um Editor de Regras de negócio (Figura C.1) que é suportado por repositórios de vocabulários, regras e ontologias de serviço. Assim, para facilitar o desenvolvimento de regras de negócio, o Editor de Regras baseia-se na incorporação e uso de modelos (*templates*) associados aos diferentes tipos de regras de negócio.

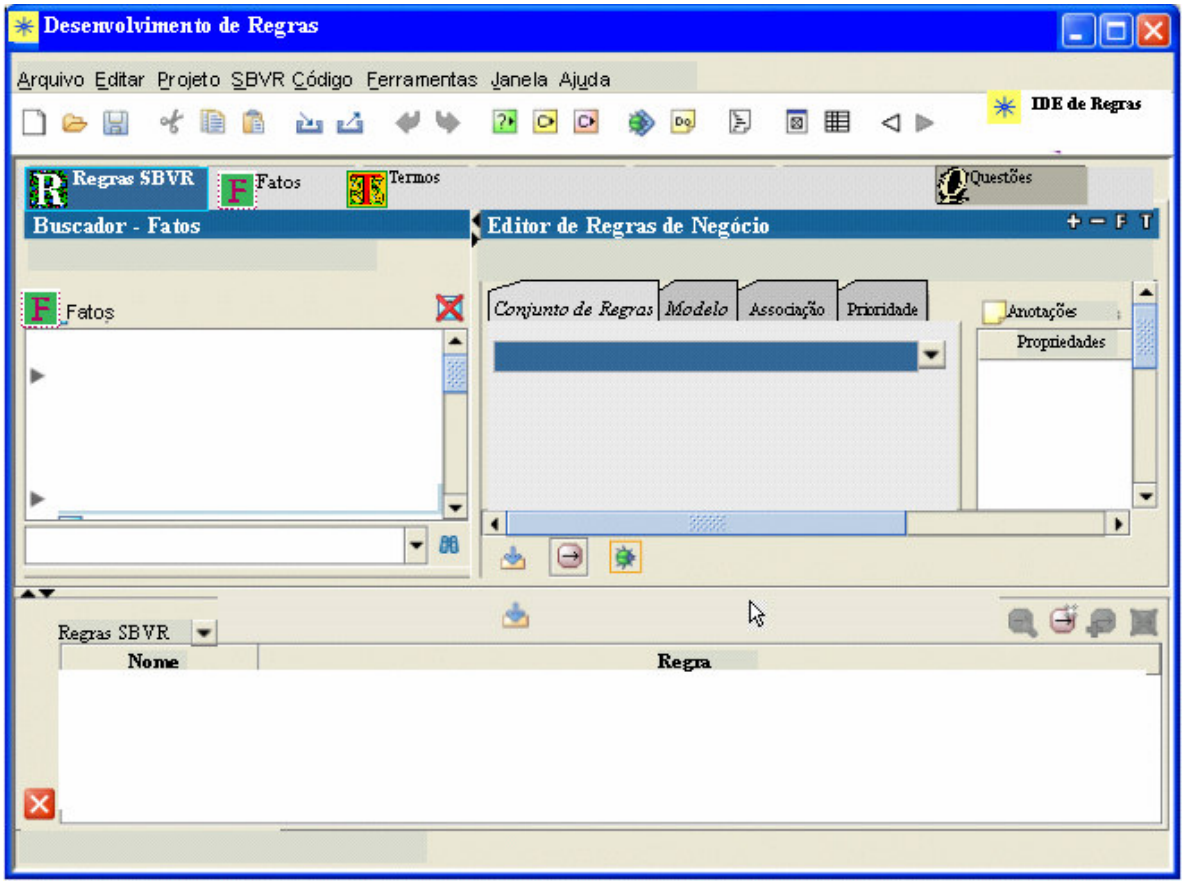

Figura C. 1 – Interface do Editor de Regras.

Para criar as regras bem como os fatos e termos que compõem as regras o analista de negócio deve antes de tudo, selecionar no Editor o vocabulário da comunidade alvo. Para isto deve clicar em (Arquivo→Abrir) ou clicar no ícone e escolher o vocabulário em questão. Supondo que o vocabulário escolhido é o identificação.rul, da comunidade de identificação civil, a tela do Editor de Regras ficará conforme mostra a Figura C.2.

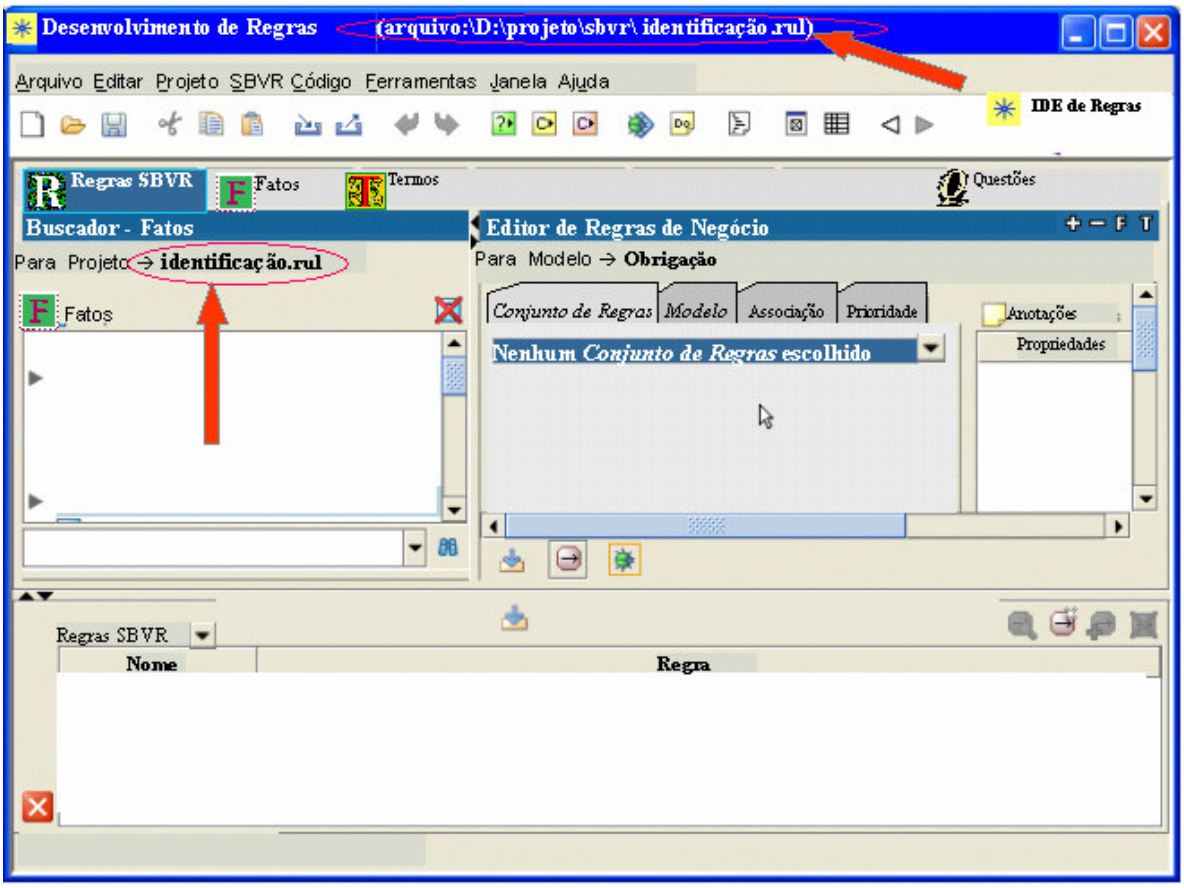

Figura C. 2 – Editor de Regras com vocabulário "identificação.rul" aberto.

Feito isto o analista de negócio deve escolher o conjunto de regras no qual trabalhará, escolhendo a ficha "Conjunto de Regras" e selecionando um dos conjuntos de regras na lista apresentada (Figura C.3).

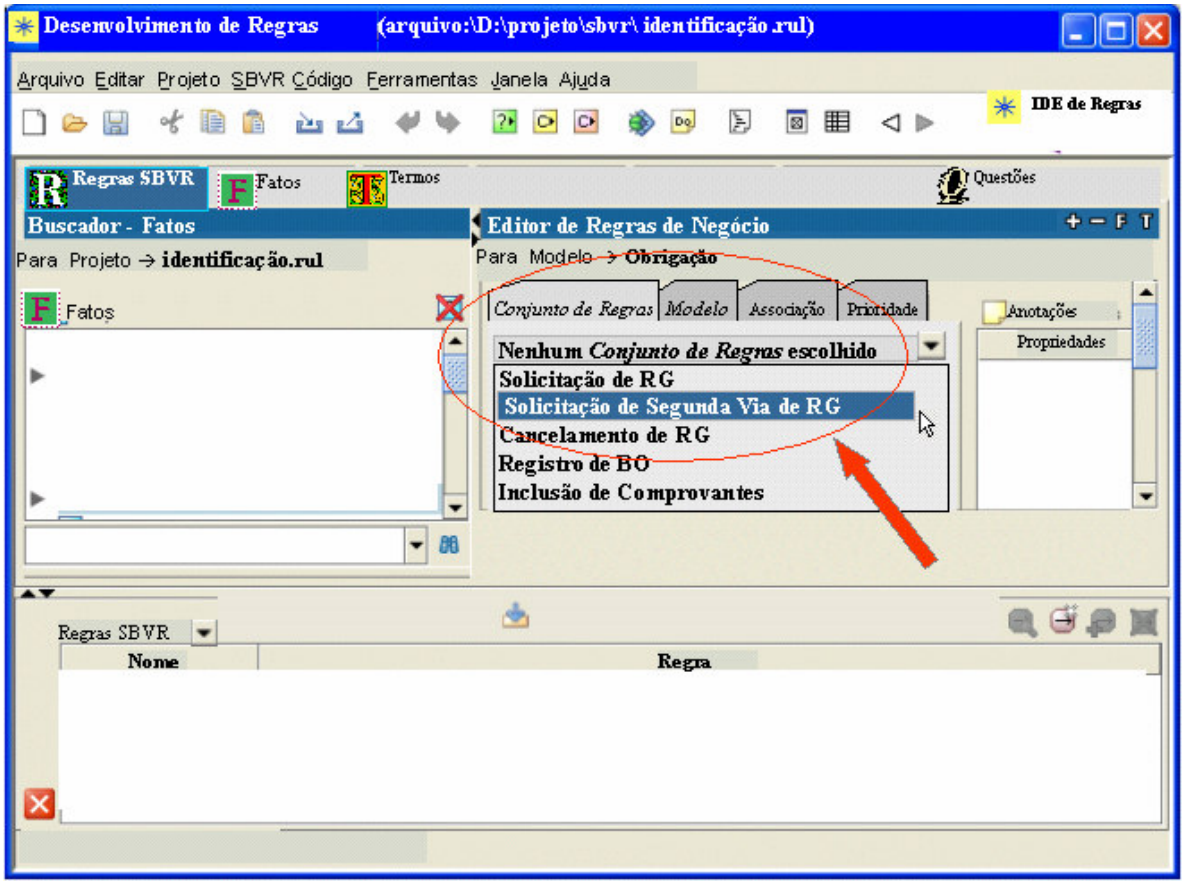

Figura C. 3 – Seleção de Conjunto de Regras.

As eventuais regras já definidas no conjunto de regras escolhido aparecerão no painel inferior, bem como os fatos já definidos para a comunidade aparecerão no painel à esquerda (Figura C.4).

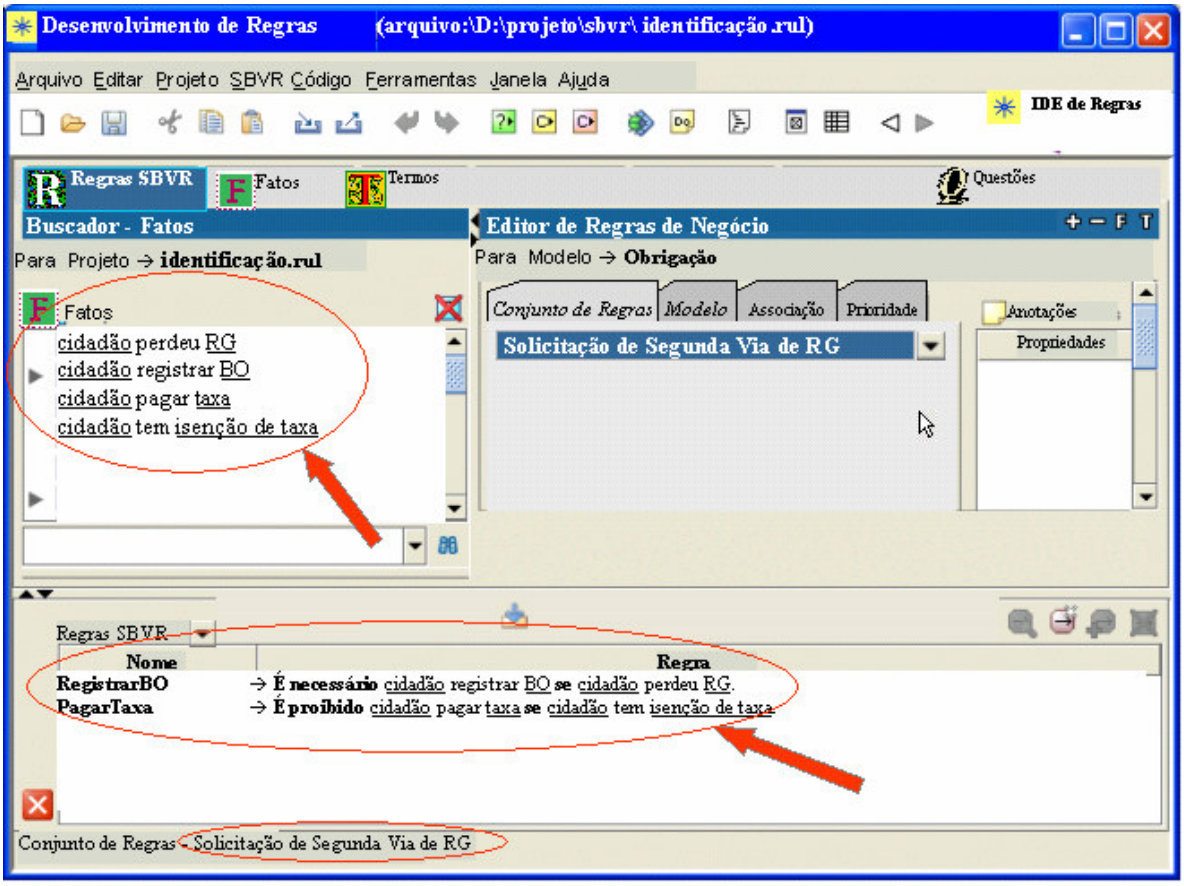

Figura C. 4 – Regras e fatos associados ao conjunto de regras e à comunidade.

Neste ponto, se o analista de negócio quiser criar uma nova regra de negócio ele deve primeiro selecionar a ficha "Modelo" e escolher um na lista que será disponibilizada (Figura C.5).

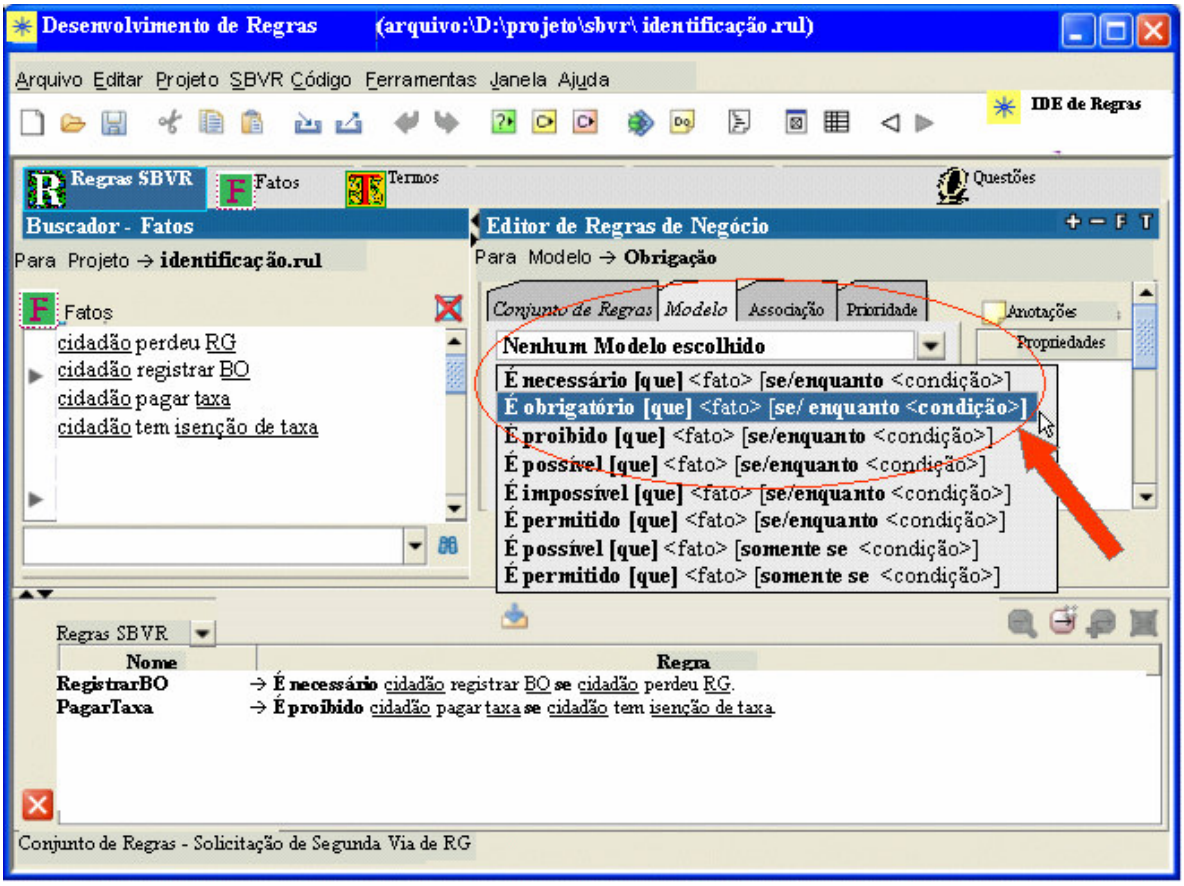

Figura C. 5 – Seleção de Modelo de Regra de Negócio.

O analista de negócio seleciona o modelo desejado e clicando-se sobre <fato> ou <condição> no modelo em edição o Editor fornece a lista de predicados possíveis naquele contexto, conforme mostra a Figura C.6. Por exemplo, no modelo **"É obrigatório** [**que**] <fato> [**se/enquanto** <condição>]", apontando-se para <fato> ou <condição> busca-se a lista de predicados relativos ao vocabulário selecionado. Uma vez escolhido o predicado e apontando-se para um de seus campos busca-se no vocabulário a lista de termos que fazem sentido naquele contexto.

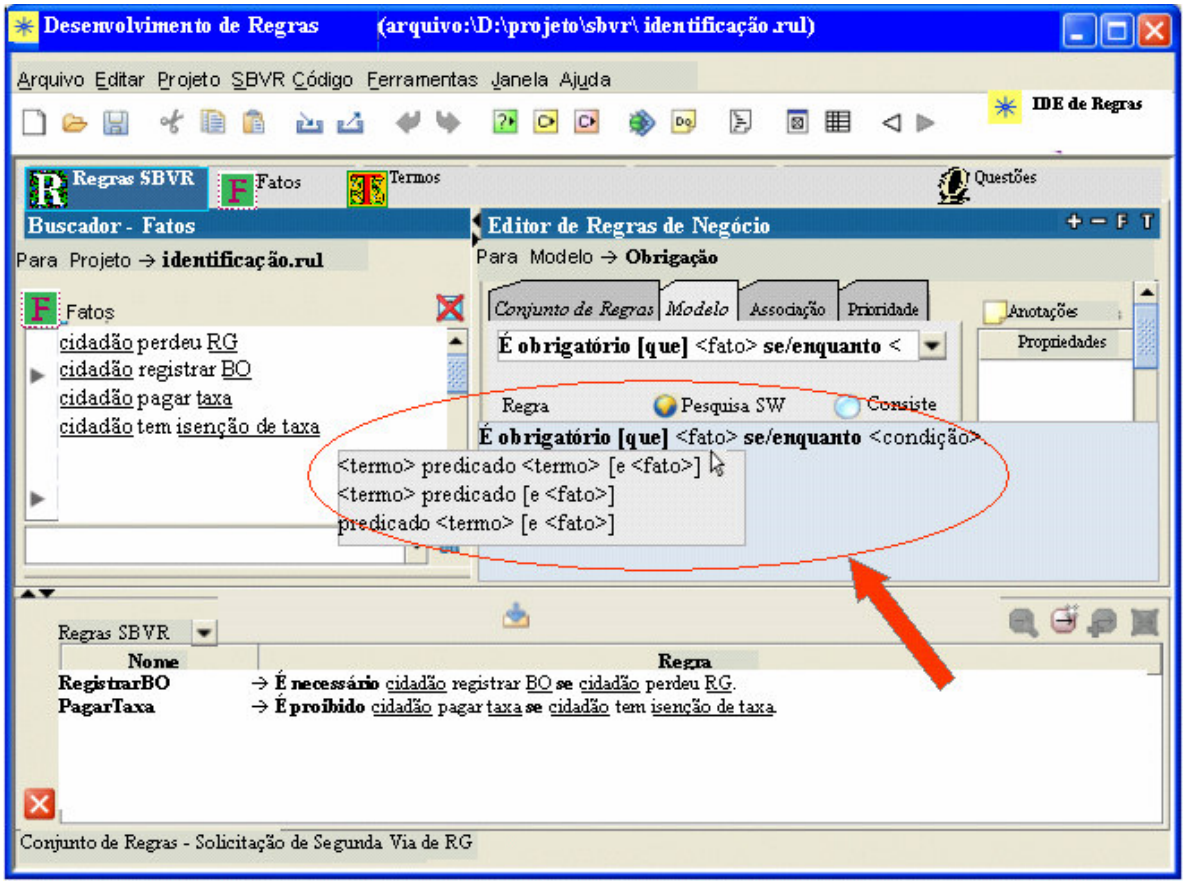

Figura C. 6 – Edição de Modelos de regras com facilidades para incluir fatos.

Outra alternativa para incluir fatos num modelo de regra é selecionar um fato no painel à esquerda do Editor e arrastá-lo para a posição <fato> ou <condição> no modelo. Os eventuais termos, verbos ou predicados ainda não definidos quando da definição de uma regra serão incluídos automaticamente no vocabulário em questão. A Figura C.7 mostra novos fatos e regra incluídos e uma nova regra em criação.

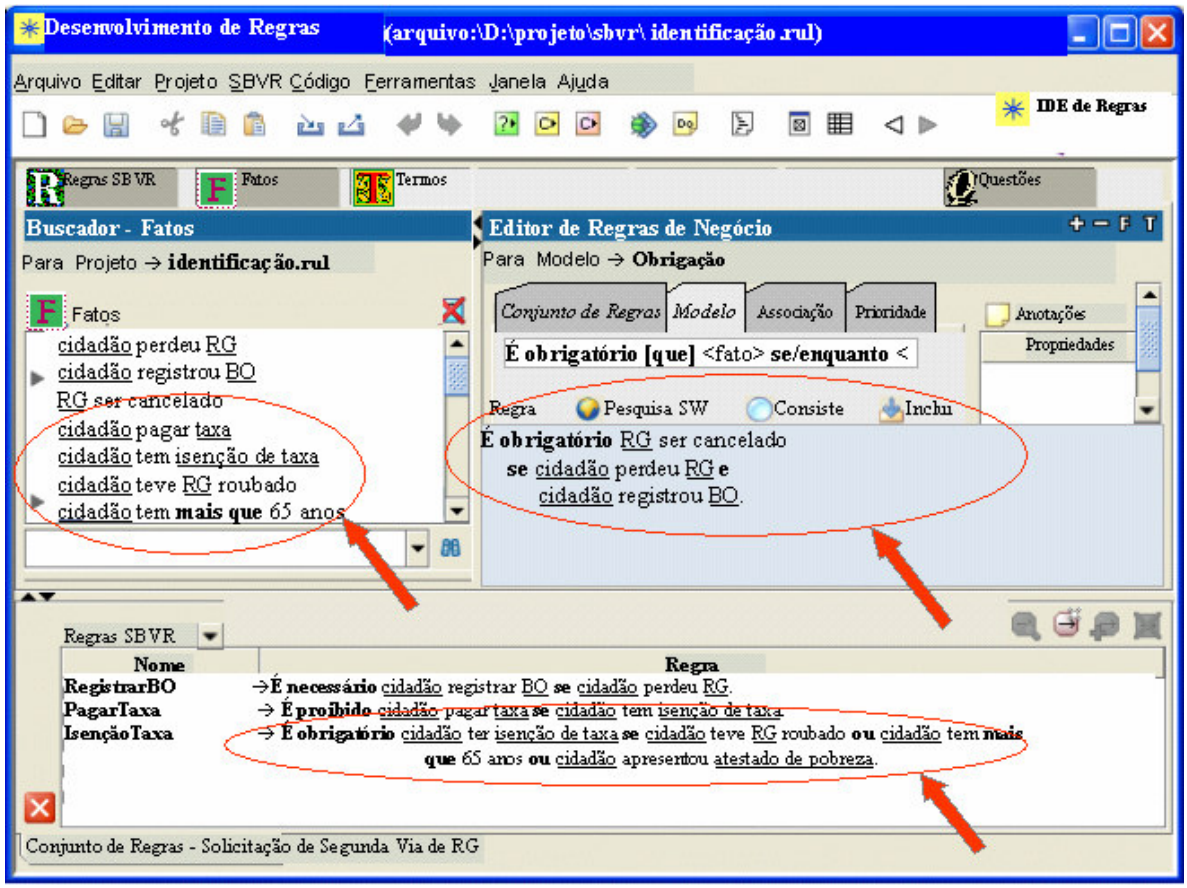

Figura C. 7 – Edição de Regra com novos fatos incluídos automaticamente.

Neste ponto, o analista de negócio pode selecionar o ícone Consiste para verificar a consistência e a sintaxe da regra em edição.

Alguns <fatos> e <condições>, para se tornarem verdadeiros, pressupõem a realização de alguma ação, que pode ser realizada por um serviço Web. O analista de negócio pode, então, selecionar o ícone Pesquisa SW para pesquisar se há Serviços Web que realizam as ações da regra. No caso da regra "Cancelar RG" em edição, pesquisa-se no Mecanismo de Registro por URLs de Serviços Web para fazer o "Cancelamento de RG" e para fazer o "Registro de Boletim de Ocorrência". Para cada URL de arquivo WSDL encontrado busca-se o respectivo arquivo WSDL. Cada arquivo WSDL contém os elementos-chave, tais como, o URL do serviço Web e os nomes das operações. Estes elementos são inseridos na Ontologia de Serviço ao qual a regra em definição deve ser associada.

Por fim, o analista de negócio seleciona o ícone Inclui para incluir a regra em edição no conjunto de regras que está sendo tratado.

C.2 Execução de Serviços

Considere um portal Web de governo que, baseado em eventos de vida, começa a tratar as necessidades do cidadão com orientações como as descritas na Figura C.8.

Figura C. 8 – Exemplo de portal de governo com facilidades de interação direta.

Supondo que o cidadão entre com "Eu perdi meu RG", o sistema determinará que o serviço se refere à "perda" de "RG" e pertence ao domínio da comunidade de identificação civil. A partir disto, o sistema terá elementos para prover as seguintes novas orientações (Figura C.9):

Figura C. 9 – Exemplo de interaçõe no portal de governo.

Assumindo-se que o cidadão entre com "José Silva" como nome, "123" como número do RG e "São Paulo" como estado emissor para a Orientação 2 e responda "sim" na Orientação 3, o sistema terá elementos para concluir que o desejo do cidadão "José Silva" é a realização do serviço "solicitar a segunda via do RG que foi emitido no estado de São Paulo" e a razão para solicitar o serviço é a "perda do RG".

Uma vez identificado o serviço, para verificar a disponibilidade de eventuais serviços Web, o sistema obtém a ontologia que refere-se ao serviço. Basicamente a ontologia contém informações que identificam a ontologia, o Serviço Web que inclui a operação que realiza o serviço, a operação em si, a URL para arquivo WSDL do serviço e os nomes do Serviços Web que colaborarão com a execução do serviço acompanhados da sua URL de arquivo WSDL e das respectivas operações. Os nomes das operações contidos na ontologia determinarão os *proxies* necessários para invocar as operações nos respectivos serviços Web.

O sistema fará também a verificação da consistência dos dados fundamentais, que consiste no mapeamento dos dados obtidos até então nos parâmetros dos serviços Web. Por exemplo, considerando-se que seja necessário realizar o cancelamento do RG, o número do RG é um parâmetro fundamental e, portanto, é necessário que entre os dados solicitados ao

cidadão esteja o número do RG, que, no caso, para o cidadão "José Silva" é o número "123".

Terminada a fase de preparação e verificação da exequibilidade do serviço o sistema passará para a fase de execução, incluindo as atividades de instanciação de máquina de regras, tradução de regras, geração de programas-clientes e a execução do serviço propriamente dita. Esquematicamente, esta fase pode ser ilustrada pelas atividades mostradas na Figura C.10.

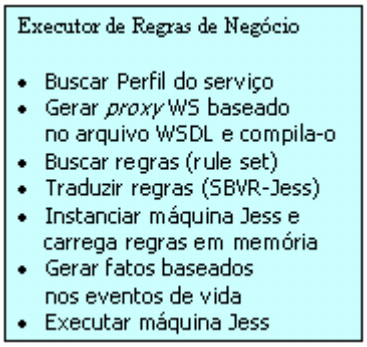

Figura C. 10 – Atividades do Executor de Regras.

Como pode ser visto, a primeira atividade é buscar o Perfil de serviço que contém informações relevantes e suficientes para a realização do serviço, tais como as mostradas na Figura C.11.

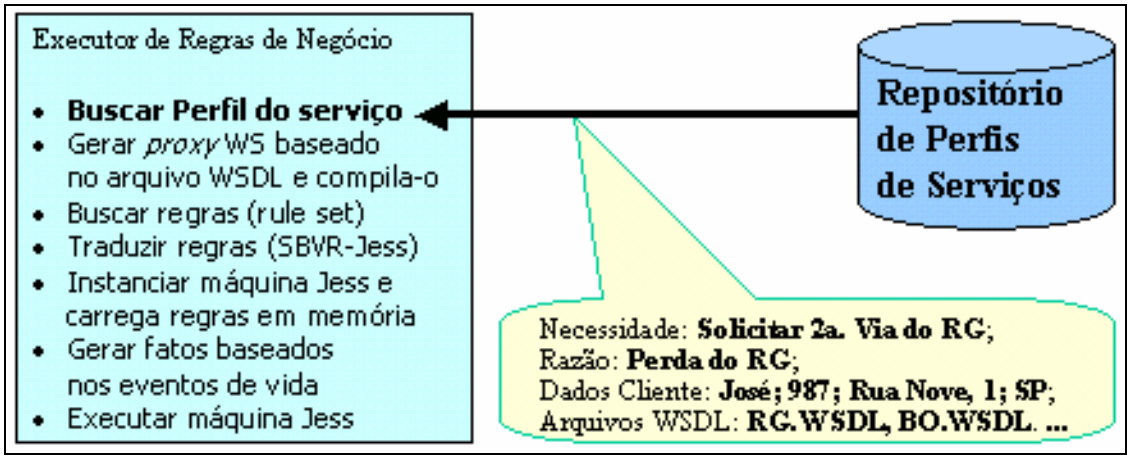

Figura C. 11 – Perfil de Serviço - Solicitar Segunda Via de RG.

A tecnologia de Serviços Web possibilita que arquivos WSDL possam ser obtidos a partir da transformação de programas Java em Serviços Web. Em outras palavras, supondo-

se que a classe Java RG, da Tabela C.1, seja transformado em Serviço Web, então este Serviço Web terá um arquivo RG.WSDL semelhante ao mostrado na Tabela C.2.

Tabela C. 1 – Classe Java RG.

```
public class RG {
  String numeroRG = "987";
  String nomeCidadao = "José";
  String status = "Nada consta";
  boolean estaCadastrado = true;
  public RG() {
  }
  public String solicitar2ViaRG() {
     return status;
  }
  public String consultarRG() {
     return status;
  }
  public String cancelarRG
              (String numeroRG) {
     status = "Cancelado";
     return status;
  }
```

Tabela C. 2 – Arquivo RG.WSDL para Serviço Web RG.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://proj_rg3" xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://proj_rg3" xmlns:intf="http://proj_rg3"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" …
  <wsdl:message name="cancelarRGRequest">
    <wsdl:part name="numeroRG" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name=" cancelarRGResponse">
    <wsdl:part name="cancelaRGReturn" type="xsd:string"/>
  </wsdl:message>
  …
  <wsdl:portType name="RG">
    <wsdl:operation name="cancelarRG" parameterOrder="numeroRG"> …
  </wsdl:portType>
  <wsdl:binding name="RGSoapBinding" type="impl:RG">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/> …
  </wsdl:binding>
  <wsdl:service name="RGService">
    <wsdl:port binding="impl:RGSoapBinding" name="RG">
      <wsdlsoap:address location="http://localhost:8080/simpleservice/services/RG"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

A geração de programas-clientes (*proxies*) é feita a partir dos arquivos WSDL dos respectivos Serviços Web e de informações contidas no Perfil de serviço. Por exemplo, para gerar o programa-cliente que chama a operação "cancelar RG" usa-se informações do arquivo RG.WSDL do Serviço Web RG e dados do cliente.

O trecho de código Java da Tabela C.3 é o resultado da geração do programa-cliente a partir desse arquivo RG.WSDL combinado com informações do Perfil de Serviço. Pode-se ver que a geração baseou-se no mapeamento de informações importantes (em **negrito**, nas tabelas), tais como, o nome do serviço, os nomes da operação, os nomes dos parâmetros de cada operação e a URL do local onde efetivamente reside o Serviço Web e os dados do cliente.

Tabela C. 3 – Programa-cliente gerado.

```
public class ProxyRG {
   . . .
 public static void main (String [] args) {
  String endpoint ="http://localhost:8080/simpleservice/services/RG";
  Service service = new Service();
  Call call = (Call) service.createCall();
  call.setTargetEndpointAddress(new java.net.URL(endpoint));
  String numeroRG = "987";
  Object[] inputParams = new Object[1];
  inputParams[0] = (Object) numeroRG;
  call.setOperationName(new QName("http://proj_rg3", "cancelarRG"));
  Object resp =  (Object) call.invoke(inputParams);
   . . .
```

Todas as regras do conjunto de regras associado ao serviço a ser executado são recuperadas do Repositório de Regras. Assim, para o exemplo da segunda via de RG, o conjunto de regras "Solicitação de Segunda Via de RG" é recuperado, contendo as regras, tais como as mostradas na Figura C.12.

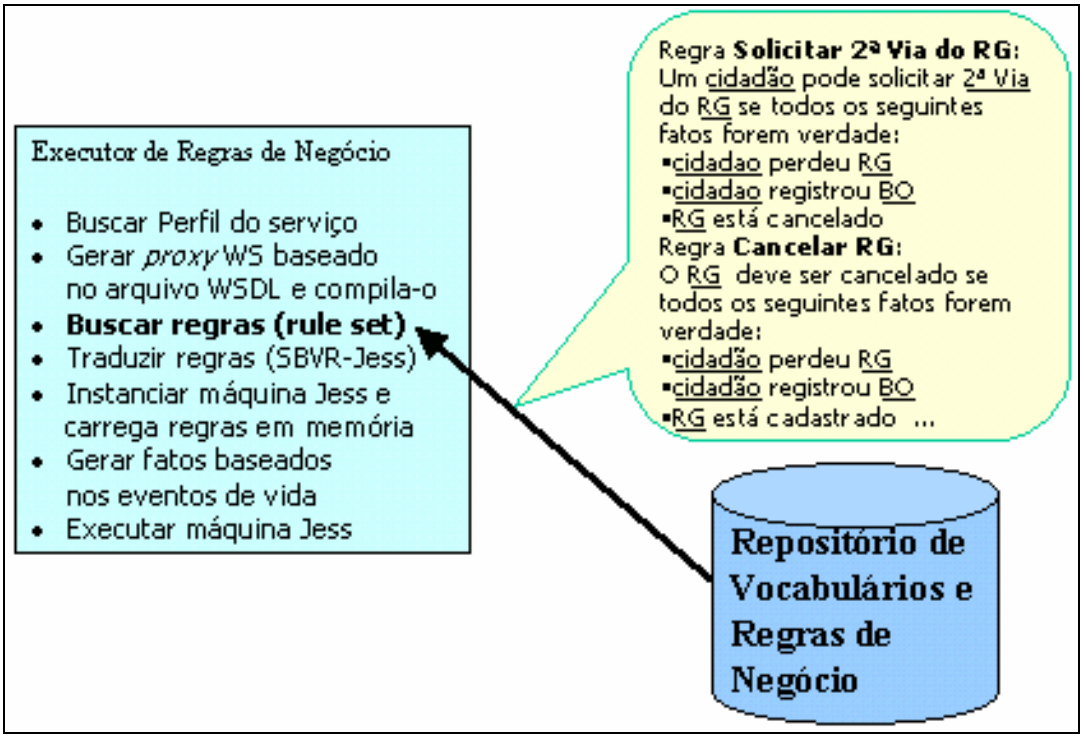

Figura C. 12 – Conjunto de regras - Solicitação de Segunda Via de RG".

As regras em notação SBVR são traduzidas para a máquina de regras alvo, que no caso é a máquina Jess. A Figura C.13 mostra um trecho da tradução do conjunto de regras "Solicitação de Segunda Via de RG" para Jess.

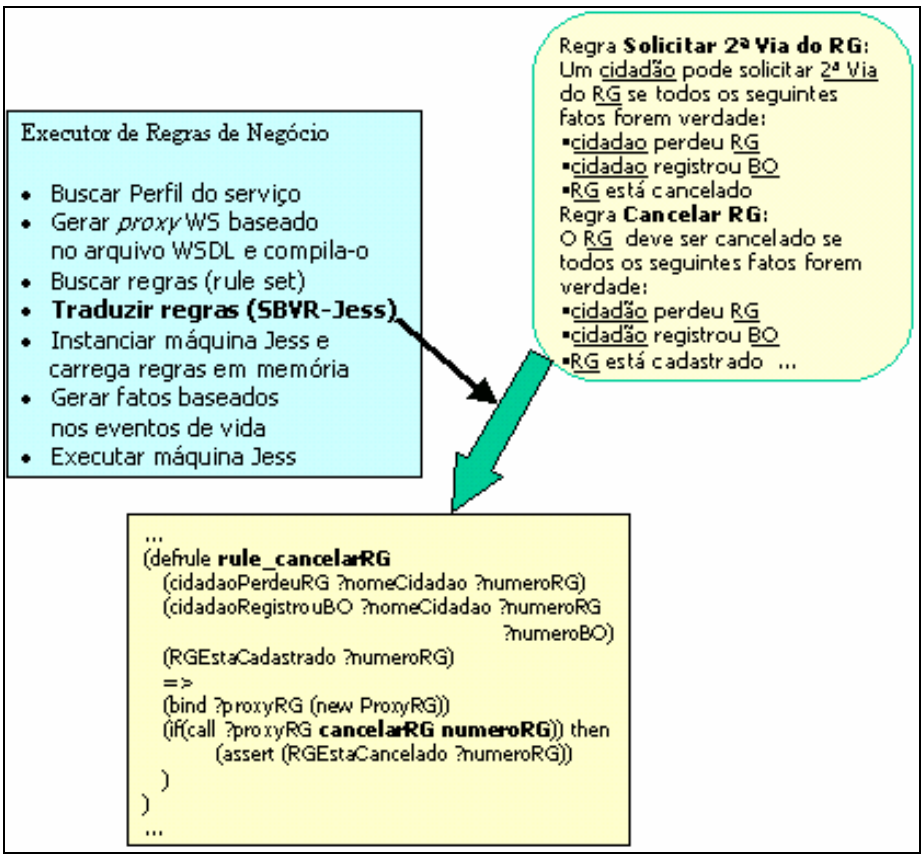

Figura C. 13 – Trecho da tradução de regras SBVR para Jess.

Estas regras traduzidas são carregadas na memória da máquina de regras, no momento de instanciação da máquina, conforme ilustra a Figura C.14.

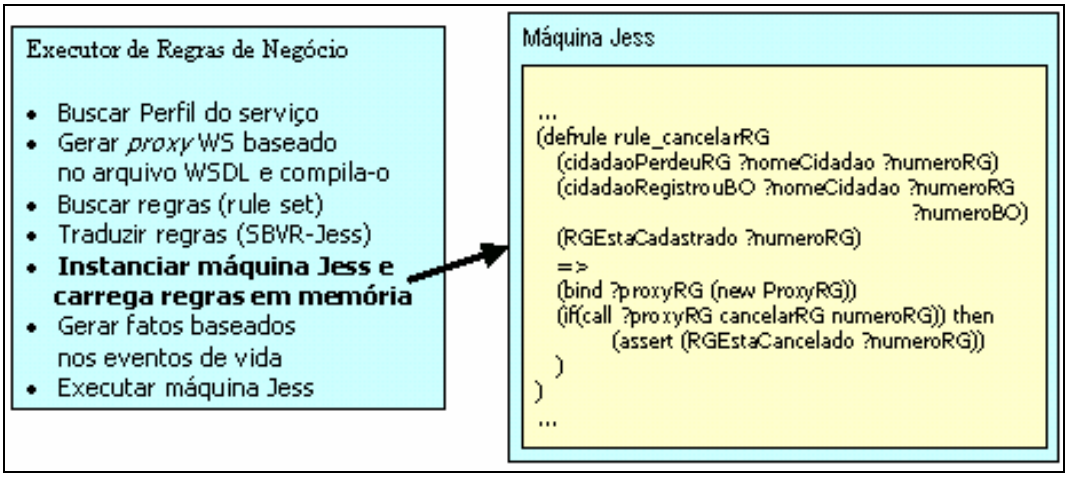

Figura C. 14 – Instanciação da máquina com carga de regras.

Feito isto, é necessário gerar na memória de trabalho da máquina de regras os eventuais fatos oriundos a partir de eventos de vida ou das razões que embasam a necessidade de realização do serviço. Um exemplo de fato é aquele oriundo da razão que embasa a necessidade de realização do serviço de "Solicitação de Segunda Via de RG", ou seja, o fato de que o "cidadão José perdeu o RG de número 987", conforme ilustrado na Figura C.15.

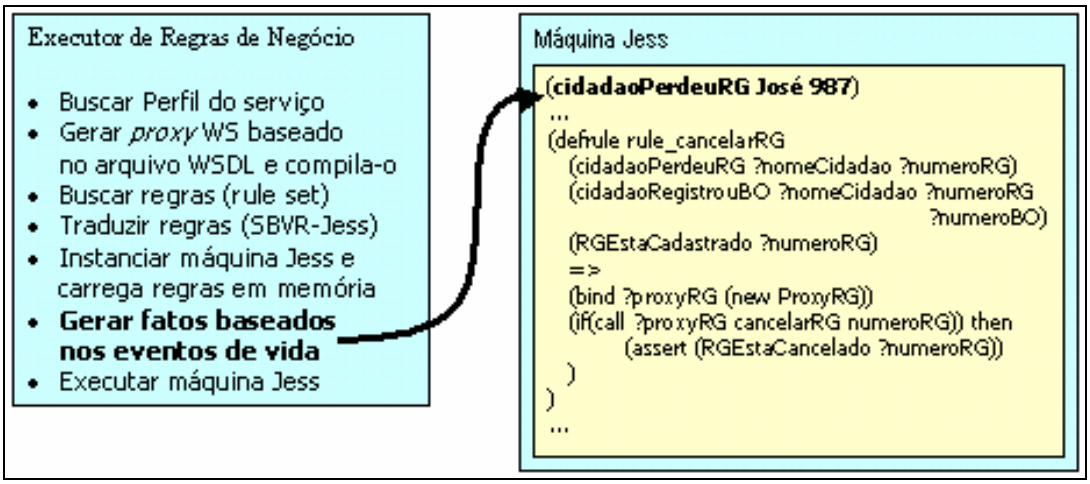

Figura C. 15 – Geração de fato baseado em evento de vida.

Ato contínuo, executa-se a máquina de regras com todas as regras e fatos carregados em sua memória de trabalho. Esta execução disparará a execução de regras. A execução das regras de negócio, por sua vez, gerarão novos fatos, que vão disparar a execução de outras regras. Algumas destas regras terão como ação a chamada de serviços Web, que individualmente realizarão parte do serviço e no conjunto realizarão o serviço desejado pelo cidadão. A Figura C.16 ilustra a execução da máquina de regras, cuja regra "rule_cancelarRG" faz chamada ao "proxyRG", que é a instância do programa-cliente anteriormente gerado.

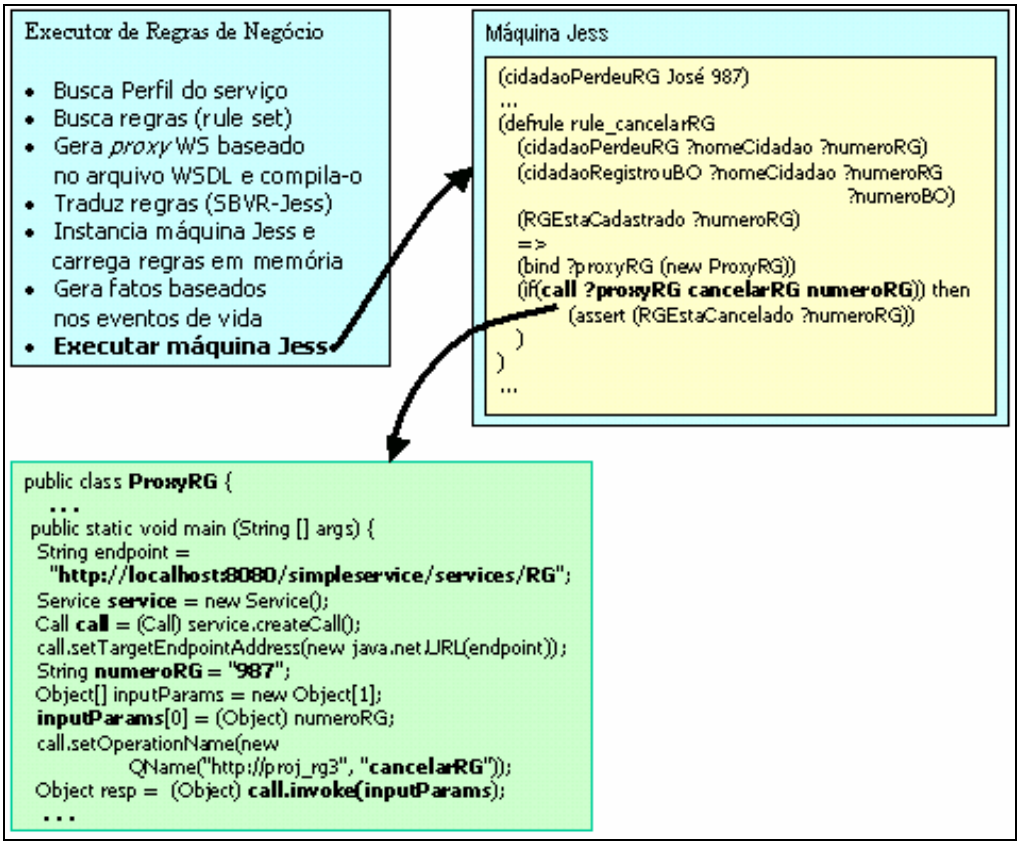

Figura C. 16 - Execução da máquina de regras com chamada de *proxy*.

A execução de "proxyRG", num dado instante invocará o Serviço Web RG (identificado pelo parâmetro *endpoint*) tendo como parâmetros a operação "cancelarRG" e o "numeroRG". A Figura C.17 ilustra esta situação.

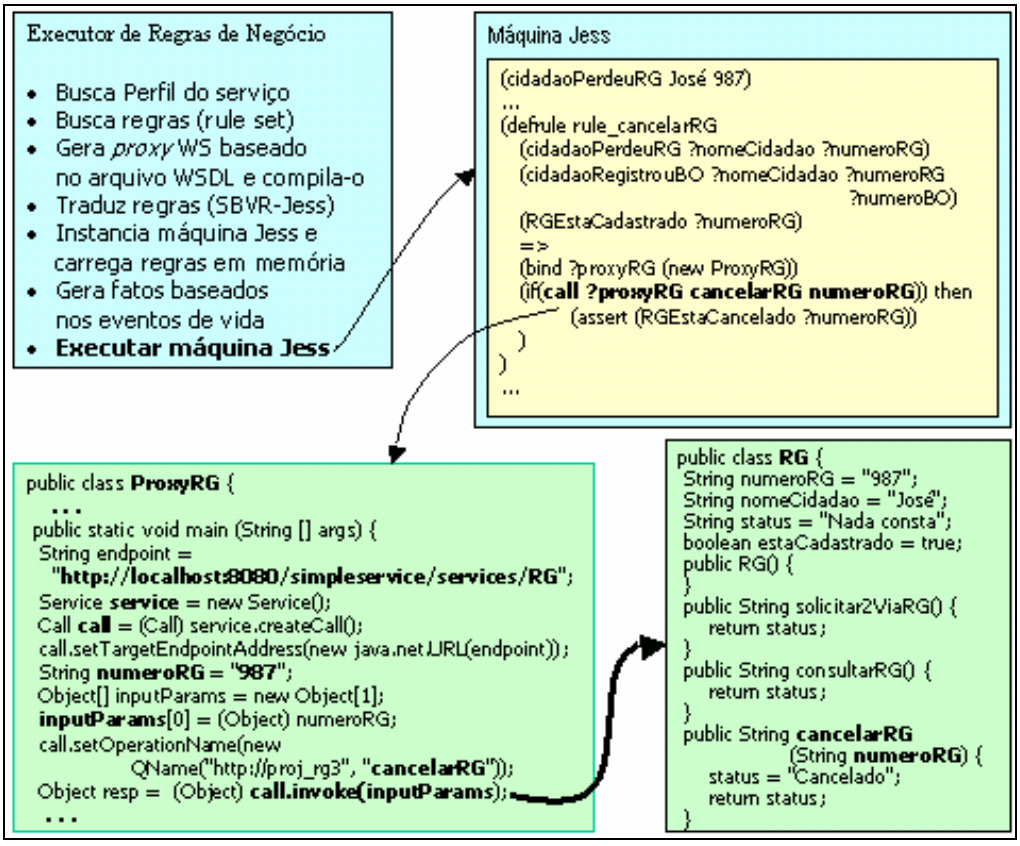

Figura C. 17 – ProxyRG invocando o Serviço Web RG.

Caso alguma das condições (*patterns* do lado esquerdo da regra Jess) não for satisfeita, uma outra regra será disparada valendo-se do mecanismo de encadeamneto regressivo de Jess. Por exemplo, na regra de cancelamento de RG (rule_cancelarRG) mostrada na Figura C.17, considerando-se que o "cidadão **não** registrou o BO" (ou seja, o pattern "cidadaoRegistrouBO ?nomeCidadao ?numeroRG ?numeroBO)" não se satisfaz), uma outra regra que trata do registro de BO será disparada e, se a ação realizada pelo correspondente serviço Web for concluída com sucesso, ocorrerá a geração do novo fato "cidadao_registrou_BO". Ato contínuo, a regra anterior, de cancelamento de RG, será agora disparada e a ação de cancelamento de RG, realizada pelo respectivo serviço Web, possibilitará a geração do novo fato "RG_foi_cancelado".