PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO

CLAUDIO OTONI MILAN DIAS

O PARADIGMA DE DESENVOLVIMENTO DE SISTEMAS BASEADOS EM REGRAS APLICADAS NA COMPOSIÇÃO DE SERVIÇOS EM UMA ARQUITETURA ORIENTADA À SERVIÇO

São Paulo

2009

CLAUDIO OTONI MILAN DIAS

O PARADIGMA DE DESENVOLVIMENTO DE SISTEMAS BASEADOS EM REGRAS APLICADAS NA COMPOSIÇÃO DE SERVIÇOS EM UMA ARQUITETURA ORIENTADA À SERVIÇO

Monografia apresentado junto ao curso de pós graduação de Engenharia de Software da Pontifícia Universidade Católica de São Paulo como requisito parcial à obtenção do título de especialista.

Orientador: Prof. Dr. André Luis Garcia Pereira

São Paulo

2009

FICHA CATALOGRÁFICA

Dias, Claudio Otoni Milan

O paradigma de desenvolvimento de sistemas baseados em regras aplicadas na composição de serviços em uma arquitetura orientada à serviço. São Paulo, 2009, 44 p.

Monografia programa de pós-graduação em Engenharia de Software – Pontifícia Universidade Católica de São Paulo. Departamento de Computação.

1. Introdução. 2. Sistemas baseados em regras. 3. Arquitetura Orientada à Serviço (SOA). 4. Composição de serviços com BPEL. 5. Mecanismos de regras de negócio (BRE). 6. Sistema para busca personalizada de produtos. 7. Conclusão. 8. Referência Bibliográfica.

Agradecimentos

Agradeço a todos meus a parentes, amigos, professores e todos que conheço e que contribuíram para meu crescimento pessoal, direta ou indiretamente.

If you have an apple and I have an apple and
we exchange these apples then you and I will still each have one apple.
But if you have an idea and I have an idea and
we exchange these ideas, then each of us will have two ideas.

George Bernard Shaw

Resumo

Este trabalho tem por objetivo estudar um método que possibilite a integração entre processos de negócio e com o processamento de regras de negócio que atendam as necessidades atuais de desenvolvimento de sistemas coorporativos. Desenvolvimento que possibilite a intervenção para melhoria e manutenção do sistema apenas por gestores do conhecimento, como os analistas de negócio. Sem grandes alterações de código, apenas diagramas e linguagens de domínio específico.

Com base em uma arquitetura de desenvolvimento de sistemas orientado à serviço, essa integração entre processos e regras de negócio será estudada em uma plataforma SOA, sobre um barramento de serviços coorporativos (ESB), onde existam processos executados em uma camada BPEL e a execução de regras de negócio sendo processadas por um BRE.

Como unificar ambas as tecnologias continua sendo o desafio a ser enfrentado neste trabalho, explorar uma possibilidade de integração entre os processos e as regras em um sistema coorporativo será o objetivo principal do trabalho.

Abstract

This work has as objective to study a method that enables integration between business process and business rules processing that attend the present enterprise system development requirements. Development that enables the improvements and system maintance by knowledge manager, business analist for instance. Without code modification, just diagrams and domain specific languages.

Based on an service oriented development architecture, this integration between process and business rules will be studied in a SOA platform, over an Enterprise Service Buss (ESB), where there are process excecution inside BPEL layer and business rules being executed in BRE layer.

The way that this tecnology could be unified still being a chalenge faced on this work, explore one integration possibility between process and rules into an enterprise system will be the main objective of this work.

Sumário

Ilustrações

Abreviaturas

BRE Business Rule Engine

BRMS Business Rule Management System

SOA Service Oriented Architecture

BPEL Business Process Execution Language

BPEL4WS Business Process Execution Language for Web Service

TI Tecnologia da Informação

BPMS Business Process Management System

WWW World Wide Web

WSDL Web Service Description Language

OASIS Organization for the Advancement of Structured Information Standards

ESB Enterprise Service Bus

EAI Enterprise Application Integration

1. Introdução

Com a grande quantidade existente de sistemas informatizados e a constante evolução dos sistemas, que exigem desenvolvimentos rápidos e contínuos, a cada dia cresce a demanda por novos paradigmas de desenvolvimento mais ágeis e flexíveis. A fim de suprir essa necessidade de desenvolvimento, conceitos como a Arquitetura Orientada à Serviço, do inglês Service Oriented Architecture (SOA), tem se fortalecido nas empresas que necessitam de grande integração entre sistemas. Tendo isso em mente, outras tecnologias estão agregando valor para o desenvolvimento e integração das lógicas de negócio aos sistemas implantados na empresa.

Dada a grande quantidade de sistemas em desenvolvimento com SOA, a evolução de técnicas para composição de serviços [1] tem se fortalecido com a Business Process Execution Language for Web Services (BPEL4WS), conhecida também por Business Process Execution Language (BPEL). No entanto, com o amadurecimento dessas tecnologias, observou-se que muitas alterações ainda demandavam mão de obra especializada em programação. Isso porque a dinâmica envolvida nos sistemas empresariais requer diversas alterações nas regras de negócio, que estavam embutidas no código fonte dos serviços.

O que vemos agora é a extração dessas regras de negócio que estavam embutidas nos códigos, para uma base externa, capaz de guardar todas as regras com baixo acoplamento ao código. Utilizando o paradigma de desenvolvimento baseado em regras, é possível extrair as regras de negócio do sistema para que este seja executado em uma arquitetura própria para o processamento de regras. Sistemas de Gerenciamento de Regras de Negócio, Business Rules Management Systems (BRMS), disponibilizam mecanismos para inferência de regras em um sistema. Com isso, é possível extrair todo o conhecimento de um sistema em regras externas ao código fonte dos sistemas. Possibilitando a alteração dessas regras de forma simples, por gestores e analistas das regras de negócio.

Entretanto, com tanta tecnologia nova envolvida no desenvolvimento de sistemas, é comum que muitas dúvidas sejam levantadas na utilização e integração de todos esses recursos disponíveis aos times de arquitetura e desenvolvimento.

Este documento servirá como uma base de referencia para trabalhos e estudos futuros, concentrando as informações necessárias para a utilização das mais modernas tecnologias e paradigmas de desenvolvimento orientado à serviço e regras. Provendo uma referencia para o desenvolvimento de sistemas baseados em regras integradas à composição de serviços de uma arquitetura orientada a serviços, utilizando para isso as tecnologias open source de desenvolvimento Java, SOA, BPEL e BRMS.

1.1. Motivação

Como definido por Krafzig (2004), o ponto chave de SOA é a prover serviços com significados concretos no nível de negócios. Por causa da relação um para um entre os mapeamentos de negócio e as entidades de tecnologia, SOA provê uma chance única, pela primeira vez na história da Tecnologia da Informação (TI), de criar valores duradouros tanto para os negócios, quanto para TI.

O interesse pelas regras de negócio por parte da comunidade de sistemas da informação vem crescendo nos últimos anos. Isso porque as regras de negócio tem se tornado um meio popular para aumentar a agilidade dos negócios. Agilidade que se refere ao contexto da agilidade dos negócios lidarem com mudanças. Regras de negócio mudam frequentemente, seja por mudanças na prática do negócio ou por novas legislações que impactam na prática de um negócio. Para se manter flexível, um negócio deve incorporar as mudanças de maneira eficiente e oportuna, segundo Ohlsson (2006).

Por conta disso, realizar o mapeamento dessas tecnologias e formalizar métodos de integrá-las é fundamental para aperfeiçoamento e utilização dela pelas empresas que necessitam de agilidade em seus sistemas de TI.

1.2. Objetivo

Por conta da crescente demanda por agilidade, este trabalho irá explorar a integração de conceitos baseados em regras sobre ambientes SOA, utilizando para isso um sistema de busca de produtos baseados em critérios do próprio cliente. Criando assim um ambiente onde a mudança de regras é uma constante e integrá-la em serviços disponíveis na arquitetura orientada a serviços é algo presente em diversos sistemas desenvolvidos atualmente.

Apesar de apresentar as tecnologias envolvidas, o enfoque principal do trabalho é estudar uma proposta de integração do paradigma baseado em regras em um ambiente SOA. Explorando uma base de conhecimento integrada à aplicação orientada à serviço. Para atingir tal objetivo, o trabalho se divide nos seguintes capítulos:

Capítulo 2 – Sistemas baseados em regras: apresenta os estudos atuais que envolvem o desenvolvimento de sistemas baseados em regras, mostrando a diferença entre ter um sistema fortemente acoplado com as regras de negócio e um sistema que externa as regras, tornando seu acoplamento muito menor.

Capítulo 3 – Arquitetura Orientada à Serviço (SOA): demonstra a base para os desenvolvimentos baseados em SOA, focando na arquitetura do sistema, que servirá como base para o desenvolvimento do sistema proposto para demonstrar o conceito de integração com regras de negócio.

Capítulo 4 – Composição de serviços com BPEL: apresenta os problemas surgidos com o crescimento dos serviços e como o BPEL tenta solucionar esta complexidade com a composição de serviços já existentes. A reutilização de serviços já desenvolvidos que se juntam para compor um novo serviço para atender as necessidades da empresa.

Capítulo 5 – Mecanismos de regras de negócio (BRE): demonstra as tecnologias existentes para o desenvolvimento de sistemas baseados em regras. Como os conhecimentos são inferidos ao sistema e de que forma o paradigma baseado em regras torna o desenvolvimento de sistemas fracamente acoplado às regras de negócio.

Capítulo 6 – Sistema para busca personalizada de produtos: nesse capítulo é apresentada a arquitetura de referencia e como pode ser feita a integração da camada de negócios com os serviços em uso.

Capítulo 7 – Conclusão: apresenta os resultados obtidos no estudo para integrar regras de negócio aos serviços de ambientes SOA, mostrando uma alternativa para o desenvolvimento de sistemas que facilite novos trabalhos.

2. Sistemas Baseados em Regras

Como constatado por Lublinsky e Tien [8], Business Rules agrega valor à implementação de atividades de negócio oferecendo um nível de flexibilidade e configuração superior, capaz de adaptar rapidamente às mudanças nos ambientes de negócio. Com o surgimento de SOA, um de seus principais enfoques está no aumento da agilidade das empresas e na diminuição do impacto das mudanças inevitáveis no sistema. No entanto, técnicas de decomposição de serviços, são capazes de criar um ambiente SOA típico onde os serviços não mudam, mas são compostos e recompostos para construir e modificar os sistemas corporativos.

Porém, a decomposição de serviços SOA não endereça as regras de negócio, que acabam sendo relacionados com os processos de negócio por conta de possuírem uma flexibilidade de alteração maior do que os componentes de SOA. Mas alguns esclarecimentos são necessários para que essas tecnologias sejam usadas como complemento, e não consideradas competidoras para o desenvolvimento de sistemas. São elas:

1. *Sincronismo*: A avaliação das regras é síncrona e um BRE é desenvolvido para avaliar as regras o mais rápido Possível. Processos, no entanto, são tipicamente de longa execução e assíncronos. O ponto forte dos processos fica por conta de sua capacidade de executar longos processos.

2. *Guardar os dados*: Regras de negócio não guardam seus dados, elas executam a partir dos dados de entrada e/ou base de conhecimento, processa e fornece o resultado (atualizando a base de conhecimento ou retornando valores) Já os processos de negócio são desenvolvidos para guardar o estado de cada instância do processo ativo.

3. *Determinismo*: Mecanismos de Regras disparam regras que calcula as condições simultaneamente, mas sem uma ordem determinística. Já os Processos de Negócio, na grande maioria,

são determinísticos e os desenvolvedores normalmente adicionam verificações para garantir que o processo seja determinístico.

4. *Granularidade*: Regras de negócio são consideradas componentes e por isso fornecem granularidade pequena e um alto nível de flexibilidade para manutenções e desenvolvimentos. Por outro lado, processos de negócios são mais estáticos e deveriam sofrer menos alterações drásticas com o passar do tempo.

2.1. Qual tecnologia usar?

Tendo em mente os pontos ressaltados, existem algumas características que devem ser levadas em conta na hora de escolher a tecnologia:

2.1.1. Orquestração de serviços

A orquestração de serviços lhe dá basicamente com atividades e serviços de longa execução ou assíncronos. Como os BREs não suportam estas necessidades, os Business Process são os mais indicados para esse tipo de implementação.

2.1.2. Componentes de negócio

Quando necessário implementar componentes de negócio, com transações instantâneas e iterações síncronas, a escolha entre regras de negócio e processos de negócio é muito difícil. Isso porque normalmente o problema pode ser resolvido com qualquer uma das tecnologias. Nesses casos os processos de negócio possuem vantagens ser de simples integração com atividades externas, através de serviços. Já as regras de negócio são vantajosas pela sua agilidade na mudança dessas regras, que podem ser feitas sem a necessidade de recompilar e implantar o processo novamente.

2.1.3. Regras de negócio na execução de processos de negócio

Existem casos onde as regras controlam também a execução dos processos de negócio, nesses ambientes é preciso considerar a complexidade das regras e sua freqüência de mudanças. Isso porque os mecanismos modernos para processos de negócio disponibilizam meios de executar regras simples dentro de um processo de negócio em execução. No entanto, a cada alteração na regra, será

necessária uma nova implantação do processo de negócio. Assim como para regras de negócio mais complexas, pode ser interessante separar a execução das regras em um novo serviço, que não necessitará de tanta intervenção para implantar as mudanças novamente. Outra aplicação das regras de negócio é em ambientes onde o processo é bem definido e estável, mas as regras sofrem mudanças constantes (como regras estipuladas pelo governo). Isso faz com que as regras de negócio suportem as mudanças dinâmicas das regras, que permitem modificar os processos de negócio sem a necessidade de mudanças ou reimplantação desses processos.

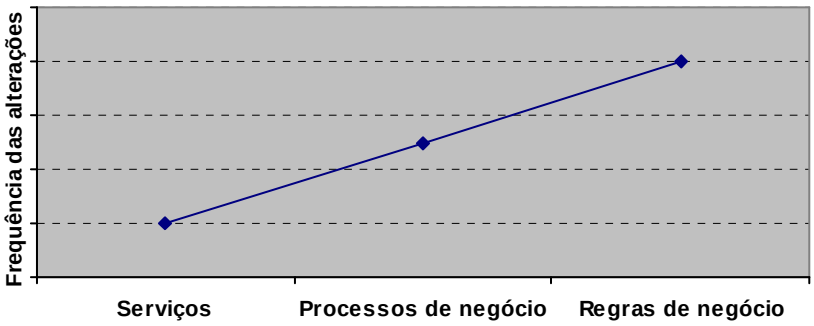

Figura 1. Freqüência de mudanças típica para serviços, processos e regras

2.1.4. Implementação SOA

A Figura 2 um uso típico dos processos de negócio e regras de negócio em uma implementação SOA.

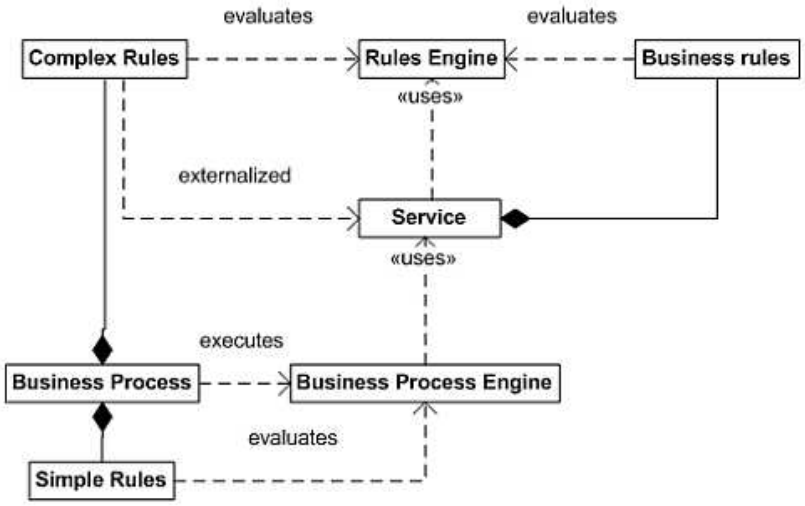

Figura 2. Implementação BRE e BPMS em um ambiente SOA

Como pode ser visto na figura acima, a implementação de processos e regras de negócio é complementar para compor uma solução SOA. Nesse caso, pode ser visto as regras de negócio como parte do serviço e os processos de negócio para a orquestração de serviços. Com as regras externalizadas como serviços de regras especiais, o processo faz as requisições de regras diretamente ao serviço de regras criado. Por conta disso, as diversas requisições feitas entre o processo e o serviço de regras podem deixar o processo com um custo caro (por conta das chamadas de rede), e por isso, Lublinsky e Tien [8] ainda ressaltam que diversos servidores de aplicação incorporam as funcionalidades de processo de negócio e mecanismo de regras.

3. Arquitetura Orientada à Serviço (SOA)

Com as necessidades de desenvolvimento ágil que os negócios demandam atualmente, a orientação à serviços surgiu como um novo paradigma para o desenvolvimento de sistemas com a intenção de agilizar o desenvolvimento e aumentar a reutilização dos sistemas desenvolvidos em uma empresa. Como mencionado por Erl (2005), orientação a serviços pode ter vários significados, mas que a lógica necessária para resolver um problema grande pode ser melhor construída, realizada e gerenciada, se decomposta em diversos problemas menores, onde cada parte representa um conceito ou uma parte específica do problema.

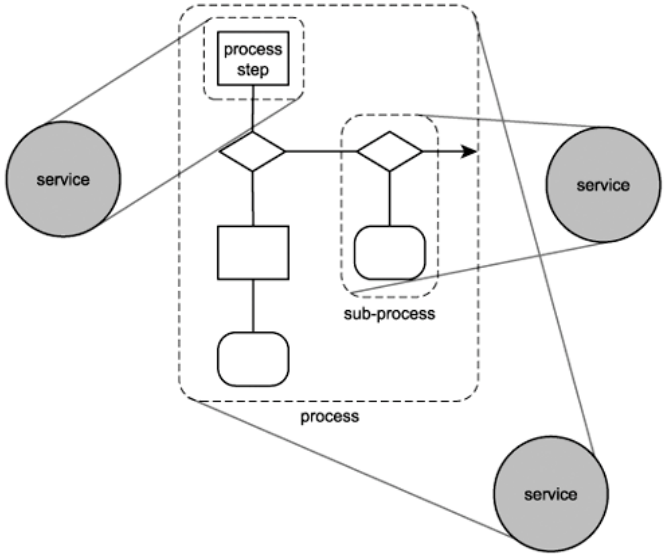

Figura 3. Serviços podem encapsular diversas lógicas do sistema

Em uma arquitetura orientada à serviços as unidades individuais são encorajadas à existir de forma autônoma, mas não isoladas umas das outras. Essas unidades lógicas devem seguir um conjunto de princípios que permite a elas evoluir de forma independente, enquanto mantêm uma quantidade suficiente de itens em comum e padronização. Em SOA, essas unidades lógicas são conhecidas como serviços, que podem encapsular diversas lógicas do sistema (). Como levantado por Parikh e Gurajada [23], alguns desafios foram no caminho para o sucesso de SOA, esses objetivos são:

1. *Alcançar visibilidade e flexibilidade nos processos:* SOA representa a fusão de diversas mudanças drásticas na computação distribuída

e possibilitou uma colaboração, como nunca vista antes, entre negócios e os grupos de TI. Provendo uma base para que as empresas ganhem visibilidade de seus dados e processos, realizem melhorias contínuas e exercitem controles finos de forma transparente e efetiva.

2. *Quebrar barreiras:* Outro objetivo de SOA é quebrar a barreira existente entre aplicações, departamentos e comercialização com parceiros que envolvem requerimentos de negócio e melhoria de tecnologia. Com seus budgets[1] anuais, cada departamento acaba construindo sua própria aplicação para resolver seus problemas diretos, com pouca visão dos demais projetos que possam utilizar as mesmas funcionalidades. Como resultado, inúmeros sistemas legados são deixados para que o departamento de TI lute para reconsolidar toda informação duplicada. Uma das promessas de SOA é quebrar essas barreiras e permitir uma melhor visibilidade dos processos e dados de uma organização.

3. *Gerenciar dados melhores:* As organizações não querem apenas gerenciar melhor seus dados, como também querem gerenciar dados melhores. É importante garantir que os dados sejam limpos, confiáveis, seguros, bem governados e rápidos. Um dos objetivos de SOA é fornecer uma plataforma de serviços de dados compostos, com um conjunto de componentes unificados para acesso aos dados, qualidade, transformação, governança e caching[2] entre os serviços.

4. *Reutilização de serviços:* Uma meta de SOA é gerenciar e reutilizar dados e serviços corporativos de maneira efetiva. Serviços desenvolvidos por um departamento pode ser utilizado por outro,

---

[1] Budget é a quantidade de dinheiro disponível para o departamento, orçamento anual.

[2] Caching é a técnica de armazenamento de dados em uma memória mais rápida para acelerar o processo de acesso aos dados.

se estiverem publicados e descritos em um formato padrão. Quando esses dados e serviços são compartilhados, o custo operacional associado à manutenção e gerenciamento torna o reuso um dos maiores incentivos de SOA.

5. *Alinhar as metas da organização:* Outra meta de SOA é alinhar os negócios e os grupos de TI a fim de desenvolver processos de negócio flexíveis e confiáveis.

Diversas confusões rondam o termo SOA, que de fato não é apenas uma arquitetura, mas sim uma metodologia, que possui diversos meios para implementar suas camadas. Parikh e Gurajada [23] apresentam uma pesquisa da AMR Research, onde pode ser vista a predominante implementação de SOA com Web services, mas outras soluções como frameworks de integração, portais, servidores de aplicação e servidores de gerenciamento de processos de negócio também estão sendo utilizados.

Existem muitos desafios para uma implementação bem sucedida de SOA, alguns deles são:

1. *Barreiras culturais:* A meta de quebrar as barreiras existentes nas organizações traz mais do que apenas desafios técnicos. As empresas precisam estar preparadas para mudanças culturais profundas nos requerimentos de uma aplicação. Isso porque os requisitos não são apenas para solucionar um caso específico, o desenvolvimento deve ser levado em conta a utilidade dessa função para toda a organização. Devem prestar total atenção nos dados e serviços que são expostos para os participantes externos ao processo.

2. *Responsabilidades:* É difícil apontar os responsáveis pelos dados e serviços em um ambiente onde uma solução implementada em um departamento pode utilizar serviços de outros departamentos. As equipes precisam seguir diversas camadas de serviços para conseguir identificar qual link está com problema e, depois de

encontrado, verificar quem será o responsável por reparar o problema.

3. *Fadiga na implementação:* Muita mudança tecnológica está envolvida em uma implementação bem sucedida de SOA. Como muitos dos sistemas legados envolvem uma complexidade muito grande, configurar tais sistemas com os processos de negócio tem apresentado uma dificuldade muito grande, que vai contra um dos objetivos de SOA.

Uma “Implementação SOA do jeito certo” é apresentada por Parikh e Gurajada [23]. Eles ressaltam que a existência de um repositório SOA com todos os recursos é o componente principal para uma arquitetura orientada à serviços bem desenhada.

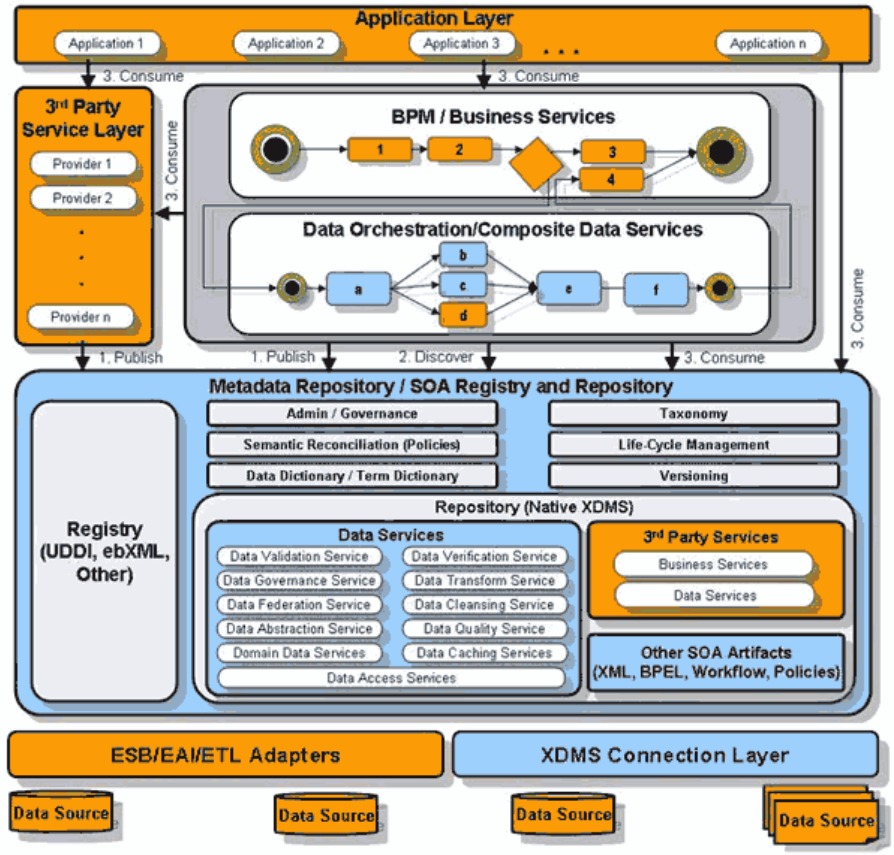

Figura 4. Arquitetura orientada à serviços bem definida

Um bom repositório deve possuir serviços de administração e governança que permita a organização dos dados e serviços de forma significativa, assim como

fornece formas de gerenciamento do tempo de vida e versionamento dos serviços. Estes serviços ficam disponíveis para as camadas de processo de negócio e workflow do repositório SOA.

A aplicação correta de SOA possibilita a eficiência, agilidade e inovação nos negócios através da informação sobre demanda, reuso dos serviços, otimização dos processos e incorporação de Web services inovadores desenvolvidos por terceiros.

3.1. Enterprise Service Bus (ESB)

O ESB é uma arquitetura de software middleware[3] que provê serviços fundamentais para arquiteturas mais complexas, muito importantes em uma arquitetura orientada à serviços. Isso porque possui mecanismos para trocas de mensagens (Figura 5), execução de transações, orquestração de serviços e realiza publicação e inscrição de funções entre funções díspares e distribuídas, segundo SearchSOA.com [22]. ESB foi desenvolvido tendo em mente suprir as falhas arquiteturais existentes nos EAIs[4], como no caso de trocas de mensagens, onde todas as aplicações eram integradas sobre um único message broker[5] que gerava um ponto único de falha. Já na criação do ESB, houve a preocupação em criar inúmeros brokers para evitar esse alto risco para sistemas de negócio complexos.

Figura 5. Integração baseada em mensagens

Tendo em vista as necessidades de SOA, as plataformas ESB possuem características que contribuem para uma implementação bem sucedida do paradigma. SearchSOA.com [22] destacou ainda que, apesar da preocupação de

[3] Middleware são programas que conectam aplicações ou componentes de software.

[4] EAI – Enterprise Application Integration são planos, métodos e ferramentas para integrar aplicações empresariais.

[5] Message broker são componentes que gerenciam as mensagens, como agenciadores.

modismo breve existente na época em que as plataformas ESB surgiram, ela tem demonstrados benefícios reais como: segurança, agilidade, disponibilidade e outras características principais de um EAI.

As funcionalidades que um barramento de serviços traz vão alem da virtualização de funcionalidades da aplicação. Com a aplicação de um ESB, não são apenas as funcionalidades de uma aplicação que são descritos pelos WSDL, a utilização de recursos como poder de processamento (CPU) e armazenamento também podem ser considerados serviços e descritos pelos WSDL. Dessa forma, os serviços podem descrever como funcionam e suas necessidades de recurso para que o barramento faça a alocação e disponibilize o necessário quando solicitado. Weerawarana (2005), destaca ainda, o importante papel dos barramentos de serviços em viabilizar uma tecnologia de processamento distribuído chamado Grid. O ESB pode ser pensado como um middleware da orientação à serviços (Figura 6).

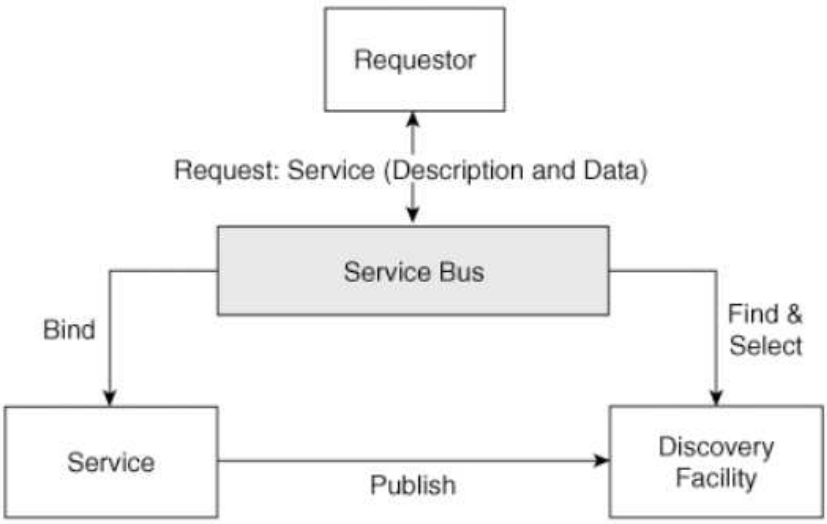

Figura 6. O barramento de serviços em uma SOA

4. Composição de serviços com BPEL

O paradigma de desenvolvimento baseado em serviços trouxe uma série de vantagens para o desenvolvimento e manutenção de soluções corporativas. No entanto, o que pode ser observado é um gap[6] entre os negócios e TI. Na tentativa de cobrir esse gap, os processos de negócio incorporam transações de negócio em múltiplas etapas e envolvem serviços automáticos e/ou pessoas, como levantado por Dias e Souza [17].

Weerawarana (2005) destaca ainda que todos os padrões já criados para disponibilização de serviços com segurança, confiabilidade e interação distribuída entre aplicações heterogêneas, podem ser vistos como infraestrutura. E que estes padrões e especificações possibilitam a criação de diversos serviços em uma rede e mecanismos para descobrir e interagir com eles. Mas que para ir alem do modelo “publicar, descobrir, interagir”, é preciso ter a habilidade de definir lógicas de interação de serviços.

Aqui pode ser visto o surgimento e aplicação de BPEL para a composição de serviços, que pode tanto suprir o gap entre os negócios e a TI, como fazer com que o desenvolvimento vá alem do modelo “publicar, descobrir, interagir”. Pois essa composição possibilita uma agregação recursiva dos serviços, isso porque ele consome os serviços publicados e gera sua própria interface WSDL[7], que pode ser usado para outras composições. BPEL foi desenhado desde o início para atender alguns requerimentos necessários para a composição de serviços.

1. *Integração flexível:* Suficientemente rico a ponto de expressar cenários onde os participantes podem ser modificados. Alem de ser capaz de se adaptar as mudanças nos serviços que estão sendo usados na composição.

---

[6] Gap é a brecha entre as duas áreas.

[7] Linguagem baseada em XML que provê um modelo para descrição de serviços web.

2. *Composição recursiva:* Disponibiliza um processo como um Web Service padrão e possibilita a composição de serviços existentes, que aumenta a escalabilidade e o reuso dos processos.

3. *Preocupação com a separação e a habilidade de compor:* Mantém a habilidade de compor dos frameworks de Web services com lógicas de composição desacopladas dos mecanismos de suporte como: qualidade de serviços, framework de mensagens e protocolos de coordenação.

4. *Conversação com memória de estados e gerenciamento do tempo de vida:* Os serviços compostos devem ter um modelo de ciclo de vida bem definido, que deve levar em conta conversações múltipla com memória de estados em processos longos com os serviços que estão se interagindo.

5. *Capacidade de recuperar:* Os processos de negócio devem ser capazes de prover um manipulador de falhas e mecanismo de compensação interno para lidar com os erros esperados que possam acontecer durante a execução.

A composição de serviços com BPEL vem sendo desenvolvida pela Organization for the Advancement of Structured Information Standards (OASIS[8]) e pode ser usada entre e internamente nas empresas. Dentro das empresas, o papel de BPEL é na padronização de integração de aplicações da empresa. Já entre empresas, BPEL facilita na interação mais efetiva com os parceiros de negócio.

4.1. Linguagem

Como BPEL foi desenvolvido como uma linguagem para a definição de processos de negócio, ela suporta dois diferentes tipos de processo de negócio:

- Processos executáveis que permitem especificar o detalhe dos processos de negócio com exatidão.

[8] OASIS - http://www.oasis-open.org

- Protocolos de negócios abstratos que permitem a especificação para troca de mensagens públicas entre os participantes.

BPEL foi construído sobre o XML e Web Services, e é uma linguagem baseada em XML que suporta as tecnologias de web service como: SOAP, WSDL, UDDI, WS-Reliable Messaging, WS-Addressing, WS-Coordination and WS-Transaction. BPEL surgiu da união de duas outras especificações anteriores, WSFL (Web Services Flow Language) e XLang. O WSFL foi desenvolvido pela IBM e utiliza conceitos de grafos diretos. Já a XLang foi desenvolvida pela Microsoft e é uma linguagem de bloco estruturado. BPEL combinou ambos os conceitos e disponibilizou um vocabulário rico para a descrição de processos.

Um processo BPEL especifica a ordem exata em que os web services serão invocados, que pode ser em paralelo ou sequencialmente. Alem disso, a linguagem é capaz de expressar comportamentos condicionais, repetições, declarar variáveis, copiar e atribuir valores, definir manipuladores de falhas, etc. Com essas características muitos podem falar que a linguagem pode ser comparada com as de propósito geral, como Java. No entanto, Juric [18] destaca que BPEL, apesar de comparável ao Java, não é tão poderoso quanto ele. Mas por outro lado BPEL é simples e mais adequado para a definição de processos de negócio. Portanto BPEL não é um substituto, mas um suplemento para as linguagens modernas como o Java.

Os processos BPEL podem ser síncronos ou assíncronos, e um processo típico primeiro recebe uma requisição, chama os web services envolvidos e responde a quem chamou o serviço de processo BPEL. Os processos síncronos bloqueiam o cliente até que o processo seja concluído. Normalmente a execução de processos de negócio utiliza os assíncronos por sua capacidade de executar processos de longa duração sem bloquear o cliente.

Para os clientes, um processo BPEL parece com qualquer outro web service. Que é a definição de um novo serviço web usado para compor outros serviços existentes. A figura abaixo apresenta um exemplo de processo BPEL.

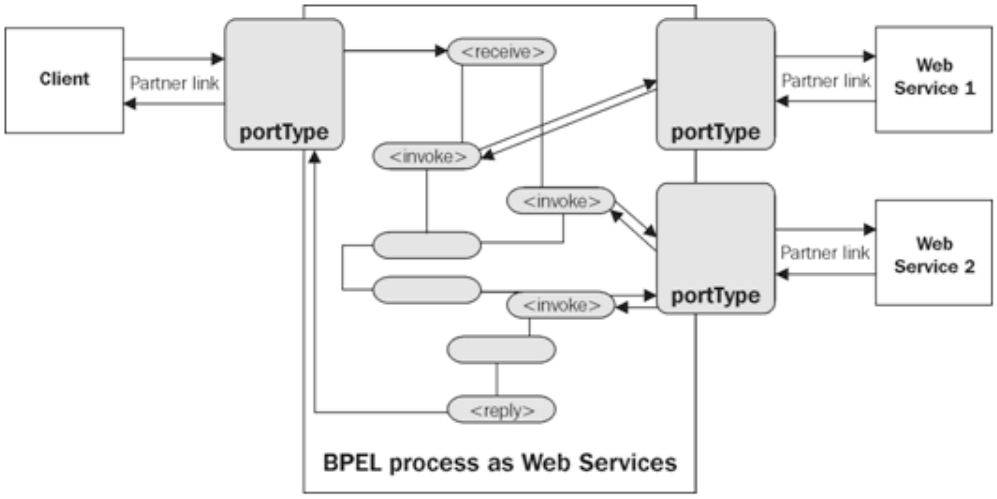

Figura 7. Exemplo de processo BPEL

4.2. Servidores

Para executar os processos BPEL é preciso de um servidor com o mecanismo de orquestração de serviços. Servidores de orquestração foram levantados por Juric [18] e provêem um ambiente para execução dos processos de negócio BPEL. Como os conceitos envolvidos na BPEL, como web services e plataformas de software moderno que suportam o desenvolvimento de web services, particularmente Java Enterprise Edition (JEE[9]) e Microsoft .NET[10].

Servidores BPEL JEE ou .NET usam seus servidores de aplicação como meio de fornecer à execução de processos serviços como segurança, transações, escalabilidade, integração com bancos de dados, componentes (como EJB[11]), sistemas de mensagem (como JMS[12]), etc.

Servidores de orquestração BPEL estão disponíveis para ambas as plataformas. Para JEE, alguns servidores são:

---

[9] JEE - http://java.sun.com/javaee/

[10] .NET - http://www.microsoft.com/NET/

[11] Enterprise JavaBeans - http://java.sun.com/products/ejb/

[12] Java Message Service - http://java.sun.com/products/jms/

- Oracle BPEL Process Manager (http://www.oracle.com/technology/products/ias/bpel/index.html)
- IBM WebSphere Business Integration Server Foundation (http://www.ibm.com/software/integration/wbisf)
- IBM alphaWorks BPWS4J (http://www.alphaworks.ibm.com/tech/bpws4j)
- ActiveBPEL engine (http://www.activebpel.org/)
- Intalio BPM (http://www.intalio.com/products/bpm/)
- JBoss jBPM (http://www.jboss.org/jbossjbpm/)

Já para a plataforma .NET um servidor disponível é:

- Microsoft BizTalk Server (http://www.microsoft.com/biztalk/)

A utilização em larga escala de BPEL depende de uma série de fatores e um deles é o desenvolvimento de ferramentas gráficas que permitam a criação de processos baseado em componentes gráficos. Isso se torna viável pelo fato de sua implementação utilizar uma linguagem de domínio específico e baseada em XML, que simplifica a criação de ferramentas gráficas. Alguns exemplos de ferramentas de desenvolvimento são:

- Oracle BPEL Designer
- IBM WebSphere Studio Application Developer, Integration Edition
- IBM BPWS4J Editor
- Vergil VCAB Composer
- Active Endpoints ActiveWebflow Designer

5. Mecanismos de Regras de Negócio (BRE)

Os BREs são responsáveis pela inferência de regras do sistema, mas antes disso, é importante saber que as regras de negócio possuem definições distintas de acordo com a visão que está sendo aplicada. Segundo o Business Rules Group [9], da perspectiva dos negócios, uma regra de negócio é a orientação que tem uma obrigação em relação à conduta, ação, prática ou procedimento dentro de uma determinada atividade ou esfera. Já da perspectiva dos Sistemas de Informação (Tecnologia da Informação - TI), uma regra de negócio é uma declaração que define ou restringe algum aspecto do negócio. Destinada a afirmar estrutura de negócios, ou para controlar ou influenciar o comportamento dos negócios. Por conta disso, sua aplicação tende atender diferentes necessidades de ambas as áreas (TI e Negócio).

A larga aplicabilidade das regras de negócio pode ser vista pelo levantamento feito pelo RuleML [10]. Citam o projeto da IBM para aplicação de regras de negócio para comercio eletrônico [11], que tem como missão o desenvolvimento de tecnologias altamente reutilizáveis para o desenvolvimento e com isso desenvolveram uma extensão de bibliotecas Java para regras de negócio e agentes inteligentes baseados em regras. Outra aplicação da inferência de regras está relacionada com a Rede Semântica (Semantic Web [12]) , que é uma extensão da rede mundial de computadores (World Wide Web – WWW) que possibilita as pessoas compartilharem seus conteúdos além dos limites das aplicações e websites, onde Berners-Lee [13] levantou a inferência de regras como sendo uma falha de design na implementação da Rede Semântica.

Por conta dessa importância no desenvolvimento de sistemas, os Mecanismos de Regras de Negócio tem recebido uma atenção especial do mercado. Tanto na evolução de produtos atuais, como a Rede Semântica, ou no desenvolvimento de sistemas coorporativos que necessitam da flexibilidade e agilidade de mudanças das regras de negócio.

Os principais componentes de um BRE podem ser vistos na Figura 8. Onde pode ser visto o repositório de regras que o sistema utilizará para inferir suas regras, existe uma série de tabelas com informações sobre a aplicação, definições

das regras de negocio e definições dos processos de negocio da aplicação. As interfaces de definição de regras são onde o usuário pode entrar com novas regras. Estas regras são capturadas pelas rotinas de geração de código, que serão responsáveis por gerar regras de negocio executável.

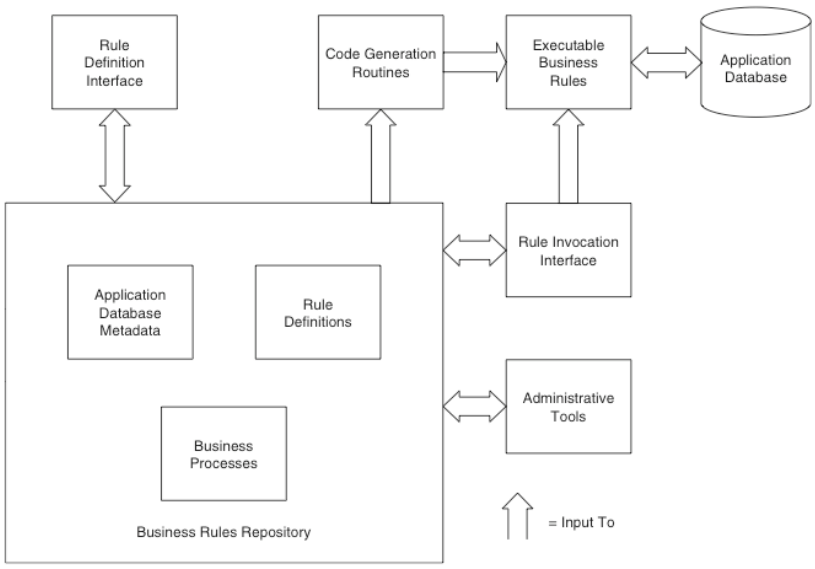

Figura 8. Componentes de um BRE

As Ferramentas administrativas adicionam recursos como gerenciamento de relatórios sobre as regras que estão sendo definidas, ao mesmo tempo em que as funções de auditoria permitem rastrear a execução da regra.

5.1. Drools Expert

Uma implementação open source de BRE é o JBoss Drools[13] Expert [14], que possui diversas características desejáveis em um BRE para implementar uma solução baseada em regras. Algumas delas são: Implementação completa do Rete, Rete seqüencial, declaração de tipos, bases de conhecimento dinâmicas para adicionar e remover regras em tempo de execução, persistência e transações, editor e depurador de regras integrado ao Eclipse[14] (Figura 9 e Figura 10), Linguagem de domínio específico (Domain Specific Language – DSL) (Figura 11) e tabelas de decisão baseados no Excel ou OpenOffice.

[13] Drools Business Logic Integration Platform – http://www.jboss.org/drools/

[14] Eclipse IDE – http://www.eclipse.com.br

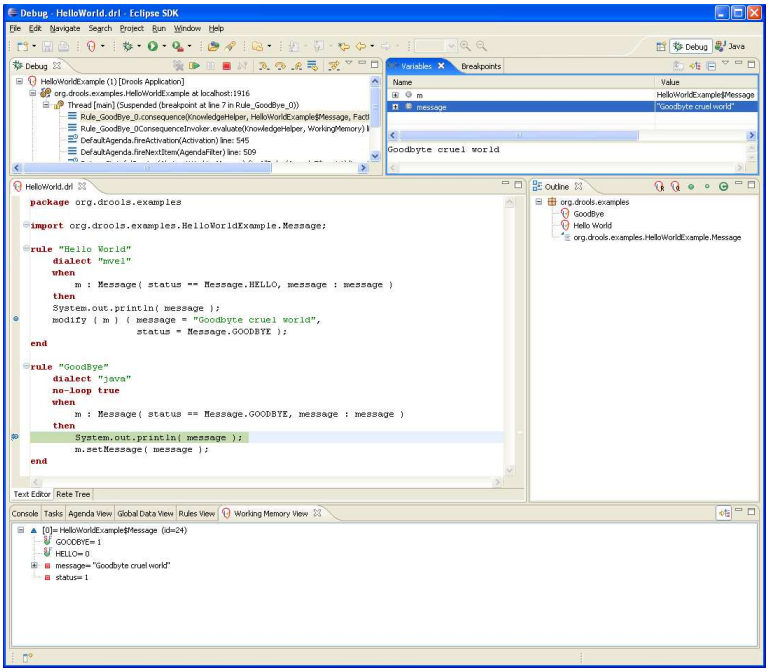

Figura 9. Depurador de regras integrado ao Eclipse

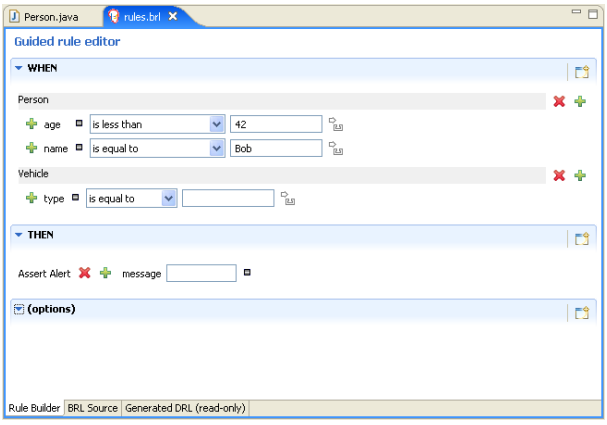

Figura 10. Editor de regras integrado ao Eclipse

```
7# Rule for Flu diagnosis
8rule "Gripe" salience 10
9    when
10       # if there is a person that has fever and headache
11       Uma Pessoa apresenta febre e dor-de-cabeca.
12   then
13       # t
14       O d
15end
16
17# Rule for
18rule "Enxaq
19   when
20       # i
21       Uma
22   then
23       # the diagnosis is Migraine
24       O diagnostico e Enxaqueca
25end
```

Uma Pessoa apresenta {sintoma1} e {sintoma2}
Uma Pessoa tem somente {sintoma}

Figura 11. DSL para programação de regras de negócio

5.2. Domain Specific Language – DSL

Para uma implementação bem sucedida das regras de negócio, é importante que o BRE possibilite o desenvolvimento baseado em alguma linguagem de domínio específico.

> *a **domain-specific language** (DSL) is a programming language or specification language dedicated to a particular problem domain, a particular problem representation technique, and/or a particular solution technique (Wikipedia [15])*

Ou seja, uma DSL é aplicada para solucionar um problema de domínio específico para que seu desenvolvimento seja simplificado e ágil. Lublinsky e Tien [8] destacam que por sua aplicação específica para os problemas de um domínio, elas capturam precisamente a semântica do domínio. Para simplificar sua utilização, uma DSL é normalmente altamente declarativa e descreve o que precisa acontecer, não o que precisa ser feito (como nas linguagens de propósito geral). Por conta disso, o ponto chave das DSL é a abstração e notações específicas do domínio, que as tornam atrativas para alguns tipos de aplicações por alguns motivos:

1. *Programação simplificada*: Por causa de seu alto nível de abstração, que é bem alinhado com o domínio do problema e define o que implementar e não o que fazer, um programa DSL são geralmente mais precisos e simples de implementar e entender.

2. *Reutilização sistemática*: DSL foca no reuso das bibliotecas que são utilizadas pela implementação da DSL. Além de que pelas DSLs serem definidas para problemas de domínios particulares, elas capturam e consequentemente reusam o conhecimento específico do domínio.

3. *Verificação mais simples*: Por conta de serem compactas e alinhadas ao domínio, uma validação de código DSL é capaz de validade que o código irá produzir resultados corretos.

4. *Aumento da cooperação*: Com o uso da mesma semântica de negócio através da organização, o compartilhamento das

informações e a redução dos mal entendidos entre as lógicas de negócio são facilitados.

5.3. Rete

"O algoritmo Rete é um eficiente algoritmo de correspondência de padrões para implementação de sistemas de produção de regras" (Wikipédia [16]), teve sua primeira publicação em 1974 e se tornou a base para diversos sistemas especialistas, incluindo CLIPS[15], Jess[16], Drools, BizTalk[17], Rules Engine e Soar.

A implementação pura de um sistema especialista deve verificar cara regra com os fatos conhecidos da base de conhecimento, disparar a regra se necessário e então mover para a próxima regra. Mas mesmo para uma base de tamanho moderado de fatos e conhecimentos, esta execução teria uma performance lenta. No entanto, os sistemas especialistas baseados no Rete constroem uma rede de nós (Figura 12), onde cada nó corresponde a um padrão ocorrendo do lado esquerdo de uma regra. Com isso, o caminho entre a raiz e as foras definem uma regra completa do lado esquerdo e cada nó tem uma memória dos fatos que satisfazem o padrão. Quando novas regras são definidas ou modificadas, elas se propagam através da rede e quando um fato ou combinação de fatos satisfazem os padrões de uma regra, um nó folha é atingido e a regra correspondente disparada.

Por isso o algoritmo foi desenvolvido sacrificando a memória em prol do aumento de velocidade. Em redes de sistemas especialistas muito grandes, esse sacrifício da memória pelo ganho na velocidade faz com que o Rete entre em problemas de consumo de memória, e por isso outros algoritmos tem sido desenvolvido para consumir menos memória.

---

[15] CLIPS Rules – http://clipsrules.sourceforge.net/

[16] Jess Rules – http://www.jessrules.com/

[17] Microsoft BizTalk Server – http://www.microsoft.com/biztalk/

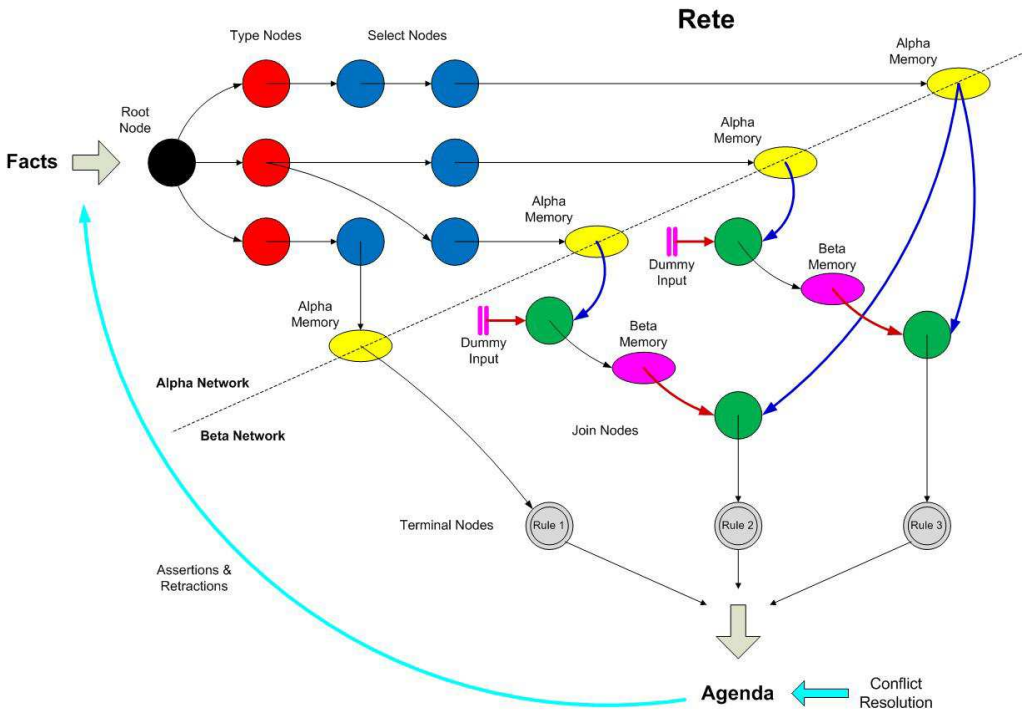

Figura 12. Diagrama de uma rede Rete

Outra característica importante do Rete é a capacidade de resolver conflitos, que durante qualquer ciclo de resolução de combinações procura por todas as combinações Possíveis para os fatos carregados na memória de trabalho. Com as correspondências encontradas, as instancias são ativadas na agenda, que determina uma ordem que as instancias serão disparadas.

6. Sistema para busca personalizada de produtos

Como visto em diversas ocasiões, o desenvolvimento de aplicações envolve conceitos não muito claros ou que possuem uma dinâmica de mudanças muito grande. Pensando nesses problemas e com as tecnologias apresentadas em mente, o desenvolvimento de um sistema para busca de produtos seguindo critérios estipulados pelo próprio cliente se enquadra perfeitamente para a aplicação de mecanismos de regras de negócio, com processos de negócio orquestrando serviços de uma arquitetura orientada à serviços.

Usando estas tecnologias, é possível criar uma arquitetura onde a manutenção e evolução do sistema será simplificada a fim de minimizar grandes intervenções de programadores. Como o intuito desse trabalho é a pesquisa de uma forma de integração entre o BRE e BPEL em um ambiente SOA, serão apresentados os requisitos do sistema, sua arquitetura básica e alguns diagramas para melhor entendimento do problema proposto. No entanto, serão assumidas algumas suposições e simplificações de uma arquitetura ou projeto ideal para que seja possível focar apenas no desenvolvimento do objeto de pesquisa e, com isso, manter um nível de abstração mais alto sobre as regras e processos de negócio para um melhor entendimento da integração BRE/BPEL.

Para a realização desta pesquisa, o desenvolvimento foi concentrado sobre as tecnologias Open Source existente atualmente e a escolha foi pelos servidores desenvolvidos pela JBoss[18]. Como estrutura de apoio para uma solução SOA, será utilizado um barramento de serviços JBoss ESB[19], JBoss jBPM[20] como BPMS e JBoss Drools como BRMS/BRE. A JBoss disponibiliza seus servidores open source gratuitamente por intermédio de diversas plataformas que podem ser integradas. Além da versão gratuita, a JBoss possui soluções completas que são totalmente integradas para implementação das tecnologias SOA comercializadas e

[18] JBoss Community – http://www.jboss.org/

[19] JBoss ESB – http://www.jboss.org/jbossesb/

[20] jBPM – http://www.jboss.org/jbossjbpm/

possuem um suporte especializado. No entanto, apesar de algumas dificuldades no início para fazer com que todos os servidores trabalhem em conjunto, as soluções open source atendem as necessidades de desenvolvimento.

6.1. Requisitos do sistema

Como o objetivo deste trabalho é o estudo das formas de integração de um BRE com BPEL, será proposto aqui uma parte básica do sistema de busca personalizada de produtos para que seja possível trabalhar com a integração nos processos que serão apresentados de forma mais clara. A Figura 13 apresenta o caso de uso dos requisitos que serão atendidos.

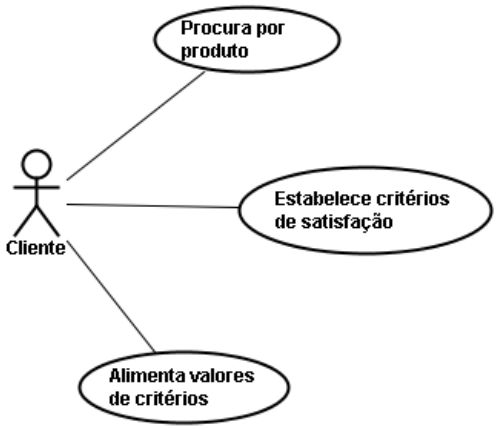

Figura 13. Caso de uso proposto

Procura por produto – neste caso de uso, o cliente entrará com o identificador do produto e o sistema deverá retornar qual é o produto mais interessante para ele de acordo com suas expectativas, cadastrada por intermédio dos critérios de satisfação.

Estabelece critérios de satisfação – onde o cliente fará o gerenciamento de seus critérios de satisfação. Dados que devem ser utilizados para encontrar o produto desejado pelo cliente seguindo todos os critérios disponibilizados ao cliente.

Alimenta valores de satisfação – o cliente utiliza esta função para informar o grau de satisfação pessoal dele com determinado critério do produto. Para que o cliente tenha a oportunidade de conhecer novos produtos, não deve existir nenhuma relevância do critério avaliado por ele mesmo.

6.2. Arquitetura

Com as tecnologias e servidores apresentados em mente, a arquitetura macro do sistema em trabalho pode ser vista na figura abaixo. Para manter a solução simples, toda arquitetura do sistema foi desenvolvida sobre um único servidor (única estrutura de hardware). Com isso, foram desconsiderados diversos requisitos não funcionais (escalabilidade, performance, segurança, redundância, etc.) que são fatores arquiteturais decisivos para o desenvolvimento de sistemas.

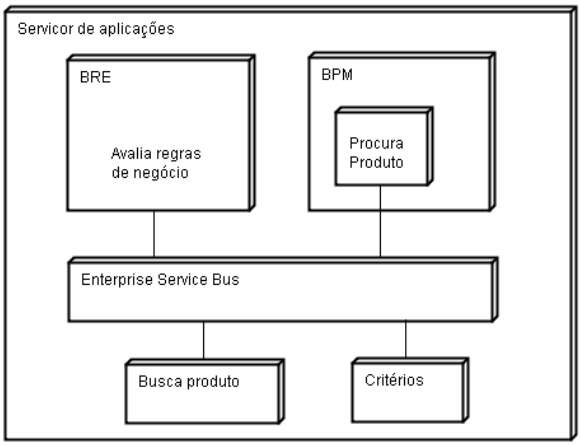

Figura 14. Arquitetura proposta

No diagrama acima, pode ser vista a relação proposta entre as regras de negócio e os processos de negócio, onde é fortalecido o baixo acoplamento entre essas duas camadas para que o usuário tenha liberdade em adicionar suas regras sem alteração alguma nos processos de negócio.

Além do baixo acoplamento entre os serviços, o ESB ainda fornece uma variedade de recursos que podem ser utilizados para incrementar a solução e criar um ambiente dinâmico de acordo com as necessidades da empresa e do sistema em desenvolvimento.

6.3. Integração

Com base na arquitetura apresentada, surgem duas possibilidades para integração dos recursos de regras e processos de negócio. A primeira delas, apresentada na Figura 15, utiliza um recurso disponível no ESB para que um serviço intercepte o outro antes ou depois de sua execução. Tal recurso é útil para transformação dos dados em casos de integração com outros sistemas. Mas nessa

aplicação, sua utilidade será para aplicar aos dados encontrados as regras de negócio criadas pelo usuário após a execução do serviço de busca de produtos. Nesse momento, os produtos disponíveis já estarão selecionados e será possível aplicar as regras para encontrar o produto de maior interesse ao cliente.

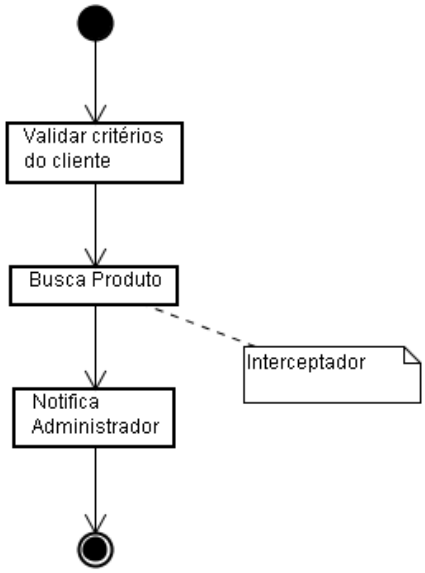

Figura 15. Processo interceptado pelas regras

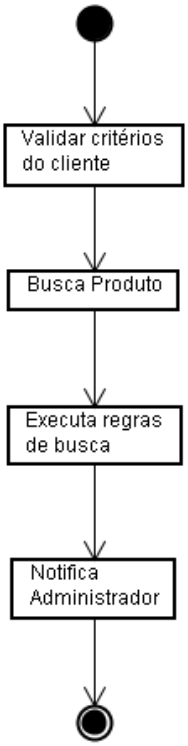

Figura 16. Processo chamando as regras

Já o segundo caso, apresentado na Figura 16, representa uma arquitetura onde as regras serão disponibilizadas ao sistema na forma de um novo serviço. Assim, qualquer serviço que deseje utilizar tal processamento de regras, pode fazer requisições ao serviço e obter o resultado com as regras aplicadas.

Ambas as técnicas possibilitam uma integração com um nível interessante de independência na aplicação. A primeira implementação está baseada no trabalho proposto por Rosenberg e Dustdar [19] e toma por base que o ESB fará a interceptação do serviço de busca de produto para processar as regras. Com essa idéia os processos e as regras ficam totalmente isolados, mas uma alteração mais drástica no processo requer alterar, não apenas o processo e regras, como também as configurações para que o ESB processe as regras no momento correto. Já para a segunda implementação, o grande benefício fica por conta da possibilidade de reutilização das regras de forma mais livre. Isso por que como elas estão expostas como um Web Service para a empresa, essa tarefa facilita muito na hora que algum outro setor precisar processar as regras sobre os dados de sua aplicação, esse reuso pode ser realizado sem alterações no ESB ou no BRE.

Com alterações em processos BPEL é possível criar ou alterar toda a rotina de solicitação dos produtos, mantendo uma única base de conhecimento inalterada. No entanto, qualquer alteração nas bases de regras deve levar em conta que diversos serviços estão utilizando elas com uma determinada interface. O que requer uma boa análise inicial para que não seja necessário alterar a interface do serviço de regras posteriormente, uma vez que essa alteração forçará um retrabalho em todos os serviços que utilizam a base de regras exposta.

7. Conclusão

O desenvolvimento de sistemas coorporativos vem crescendo a cada ano e a necessidade por um paradigma de desenvolvimento que agilize, tanto o desenvolvimento quanto as alterações e manutenções do sistema, é cada vez mais exigido pelo mercado.

Dessa forma, cada vez mais as empresas estão adotando soluções SOA que possibilitem uma maior integração entre todos os sistemas ao mesmo tempo em que viabiliza a criação de novos sistemas a partir da orquestração dos serviços disponíveis na empresa, integrando ainda com serviços de seus terceiros. Além de reaproveitar todo sistema legado da empresa com camadas de serviço que disponibilizam interfaces de acesso aos sistemas já em operação.

Sendo assim, a aplicação de tecnologias para orquestração de serviços tem se fortalecido para que a flexibilidade necessária para atender o negócio seja atendida. Uma das tecnologias que tem tomado força nesse desenvolvimento é o BPEL, que viabiliza a criação e alteração de processos com facilidade pelos analistas de negócio. Outra realidade atual, a execução de regras de negócio, tem se fortalecido para viabilizar o processamento de regras sobre dados e até mesmo para a tomada de decisões na execução dos processos.

Com isso, o estudo de formas que viabilizem a utilização de ambas as tecnologias em conjunto são fundamentais para o sucesso e aplicação no mercado. Na implementação apresentada na Figura 15 existe uma perda na praticidade de reuso das regras na forma de web services, isso porque as regras interceptam o tráfego de um processo no ESB. Para que essa interceptação funcione, são necessárias algumas configurações no ESB para que as regras sirvam como recursos de transformação pós processamento da busca produtos. Apesar de esta abordagem trazer um baixo acoplamento entre os serviços, pois alterações realizadas nas regras de negócio serão colocadas em prática sem necessidade de alteração no processo de negócio BPM, grandes alterações que envolvam mudança nas regras de negócio e nos fluxos de processos que utilizam essas regras, terão uma complexidade um pouco maior devido às configurações necessárias no ESB.

Dessa forma, os próprios analistas de negócios estarão aptos a alterar a base de conhecimento conforme alterações de requisitos. Bem como realizar alterações no processo de forma simples, tendo em mente que o ESB aplicará as regras de negócio nos dados assim que eles forem procurados pelo serviço de busca. No entanto, mudanças mais drásticas no processo podem sofrer com essa configuração do ESB ou mesmo a criação de novos processos que reutilizem as regras podem não ser uma atividade trivial.

Já a implementação apresentada na Figura 16, o acoplamento entre os processos e as regras, apesar de baixo, acaba sendo superior ao apresentado anteriormente. No entanto, essa abstração servirá para introduzir a flexibilidade de reutilização das regras e processos para compor novos serviços na empresa. Isso por que a interface de acesso às regras passa a ser uma interface padrão web service e poderá ser acionada por qualquer outro processo em execução no ESB, ou até mesmo algum consumidor de serviço externo, caso seja aplicável.

Com isso, vale à pena estudar a aplicação e reutilização de todos os componentes da arquitetura, e valorizar aquele que for de interesse para determinada aplicação. Outros estudos podem ser realizados sobre a mesma temática, enfatizando o estudo de performance entre as duas soluções. Uma vez que todos os aspectos relacionados aos requisitos não funcionais foram deixados de lado para estudo apenas da integração entre BPEL e BRE. O desenvolvimento dessas tecnologias fortalece a gama de opções disponíveis para o desenvolvimento de sistemas e se encaixam as necessidades atuais de mudanças de requisitos de sistema, bem como nas necessidades de desenvolvimento ágil e confiável.

8. Referências Bibliográficas

1. IBM. **Business Process Execution Language for Web Services version 1.1.** Disponível em: <http://www.ibm.com/developerworks/library/specification/ws-bpel/>. Acesso em: 10 jul. 2009.
2. CAMBRIDGE. **Cambridge Dictionaries Online.** Disponível em: <http://dictionary.cambridge.org>. Acesso em: 28 jul. 2009.
3. MICHAELIS. **Dicionário Online Michaelis.** Disponível em: <http://michaelis.uol.com.br/>. Acessado em 30 jul. 2009.
4. WIKIPEDIA. **Business Rules Management System.** Disponível em: <http://en.wikipedia.org/wiki/Business_rule_management_system>. Acesso em: 04 ago. 2009.
5. WIKIPEDIA. **Business Rules Engine.** Disponível em: <http://en.wikipedia.org/wiki/Business_rules_engine>. Acesso em: 04 ago. 2009.
6. OHLSSON, Jesper. **Enforcing Business Rules in E-Business Systems:** A Survey of Business Rule Engines. 2006. 43 f. Dissertação - University of Skövde, Skövde, 2006.
7. KRAFZIG, Dirk; BANKE, Karl; SLAMA, Dirk. **Enterprise SOA:** Service-Oriented Architecture Best Practices. New Jersey: Prentice Hall Ptr, 2004. 408 p.
8. LUBLINSKY, Boris; TIEN, Didier Le. **Implementation of business rules and business processes in SOA.** Disponível em: <http://www.infoq.com/articles/business-rules-processes>. Acessado em: 10 set. 2009.
9. BUSINESS RULES GROUP. **Defining 'Business Rules'.** Disponível em: <http://www.businessrulesgroup.org/defnbrg.shtml>. Acesso em: 12 out. 2009.
10. RULEML. **RuleML:** Realize your knowledge. Disponível em: <http://ruleml.org/>. Acesso em: 13 out. 2009.
11. IBM T.J. WATSON RESEARCH CENTER. **Business Rules for Electronic Commerce.** Disponível em:

<http://www.research.ibm.com/rules/home.html>. Acesso em: 13 out. 2009.

12. SEMANTIC WEB. **Semantic Web.** Disponível em: <http://semanticweb.org/>. Acesso em: 13 out. 2009.
13. BERNERS-LEE, Tim. **Rules and Facts:** Inference engines vs Web. W3 Design Issue. Disponível em: <http://www.w3.org/DesignIssues/Rules.html>. Acesso em: 13 out. 2009.
14. JBOSS. **Drools Expert.** Disponível em: <http://www.jboss.org/drools/drools-expert.html>. Acesso em: 20 set. 2009.
15. WIKIPEDIA. **Domain-specific language.** Disponível em: <http://en.wikipedia.org/wiki/Domain-specific_language>. Acesso em: 20 out. 2009.
16. WIKIPEDIA. **Rete algorithm.** Disponível em: <http://en.wikipedia.org/wiki/Rete_algorithm>. Acesso em: 23 out. 2009.
17. DIAS, Claudio; SOUZA, Alex. **Executando processos de negócio.** Disponível em: <http://techblog.mdias.com.br/2008/10/21/executando-processos-de-negocio/>. Acesso em: 03 nov. 2009.
18. JURIC, Matjaz. **BPEL and Java.** Disponível em: <http://www.theserverside.com/tt/articles/article.tss?l=BPELJava>. Acesso em: 03 nov. 2009.
19. ROSENBERG, Florian; DUSTDAR, Schahram. **Business Rules Integration in BPEL:** A Service-Oriented Approach. Vienna: IEEE Computer Society, 2005.
20. WEERAWARANA, Sanjiva et al. **Web Services Platform Architecture:** SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and more. New Jersey: Prentice Hall Ptr, 2005.
21. ERL, Thomas. **Service-Oriented Architecture:** Concepts, Technology, and Design. New Jersey: Prentice Hall Ptr, 2005.

22. SEARCHSOA. **ESB Tutorial.** Disponível em: <http://searchsoa.techtarget.com/generic/0,295582,sid26_gci1085711,00.html>. Acesso em: 20 nov. 2009.
23. PARIKH, Ash; GURAJADA, Murty. **SOA for the real world.** Disponível em: <http://www.javaworld.com/javaworld/jw-11-2006/jw-1129-soa.html>. Acessado em: 25 nov. 2009.