APLICAÇÃO DE RACIOCÍNIO BASEADO EM CASOS NO SUPORTE A DECISÃO DE UM SISTEMA WEB DE HELP DESK

Diego Joel Pilger, Marcel Hugo
Universidade Regional de Blumenau - FURB
diegojoel@gmail.com, marcel@furb.br

Resumo
Este artigo apresenta a elaboração de um sistema web de *help desk* utilizando a técnica de inteligência artificial Raciocínio Baseado em Casos (RBC). O sistema é aplicado no domínio técnico de *help desk* de uma empresa desenvolvedora de aplicações de automação comercial. Utilizando a linguagem de programação Java em conjunto com *frameworks* de desenvolvimento web, o sistema foi desenvolvido para auxiliar no suporte a decisão dos técnicos envolvidos com atendimento a clientes.

Palavras-chave: Inteligência artificial. Raciocínio baseado em casos. Java. Sistemas de suporte a decisão. Web. MVC. JPA.

Abstract
This paper presents the development of a web help desk using the technique of artificial intelligence Case Based Reasoning (CBR). The system is applied in the field of technical help desk from a company developer of business automation applications. Using the Java programming language in conjunction with web development frameworks, the system was developed to assist in decision support technicians involved with customer service.

Keywords: Artificial intelligence. case based reasoning. Java. Decision support systems. Web. MVC. JPA.

1. Introdução

O governo brasileiro tem realizado várias ações nos últimos anos para forçar a automatização de todo o sistema fiscal e tributário das empresas. Com essas mudanças, as empresas desenvolvedoras de software, em especial aquelas voltadas à automação comercial, percebem a necessidade de aperfeiçoar os seus sistemas para atenderem a essa nova demanda de exigências governamentais. Como consequência o suporte técnico requerido das equipes de *help desk* pelos usuários de sistemas de automação comercial vem aumentando consideravelmente nesse contexto.

As equipes de *help desk* geralmente prestam suporte técnico aos usuários de sistemas de automação comercial, com base em conhecimento técnico e em materiais disponibilizados pela empresa. No entanto nem sempre as informações fornecidas pela equipe de suporte são corretas, atualizadas e resolvem os problemas reais desses usuários.

Neste contexto surgem os sistemas de suporte a decisão que auxiliam no controle e aumento de acertos na solução de problemas. Os sistemas de suporte a decisão (SSD) utilizam técnicas avançadas para se aproximarem o máximo possível do resultado esperado, fazendo uso de ferramentas de inteligência artificial como o Raciocínio Baseado em Casos (RBC).

O Raciocínio Baseado em Casos é uma técnica da inteligência artificial que simula o ato humano de lembrar experiências passadas, para resolver problemas do presente. O estudo e evolução dessa técnica ocorrem principalmente no meio acadêmico, através de aplicações práticas, principalmente nas áreas jurídica, médica, e *help desk* (WANGENHEIM; WANGENHEIM, 2003).

Com a necessidade de atender uma grande quantidade de usuários espalhados geograficamente por todo o país, mantendo ao

mesmo tempo um bom serviço de suporte técnico, as aplicações web surgem como uma ótima solução para essa questão, pois possibilitam uma maior portabilidade, baixo custo com infraestrutura e independência de sistema operacional específico.

Com base no que foi exposto, foi desenvolvido um sistema web de suporte a decisão, com aplicação da técnica de inteligência artificial RBC. Esse sistema, através do conhecimento extraído de uma base de casos estruturada, consegue solucionar novos problemas relacionados a um sistema de automação comercial. Dessa maneira, auxilia os técnicos e usuários do sistema, na solução de problemas identificados, dúvidas e na tomada de decisão.

2. Tecnologias envolvidas

Para o desenvolvimento do sistema foram utilizadas técnicas de inteligência artificial e tecnologias para desenvolvimento web, as quais são descritas nesta seção. Também são apresentados trabalhos correlatos.

2.1 Raciocínio Baseado em Casos

A técnica de Raciocínio Baseado em Casos (RBC) se baseia na busca de solução para problemas atuais em soluções passadas, utilizando uma das principais características do ser humano, a lembrança.

Segundo Wangenheim; Wangenheim (2003, p.8, grifo do autor)

> **Raciocínio Baseado em Casos** é um enfoque para a solução de problemas e para o aprendizado baseado em experiência passada. RBC resolve problemas ao recuperar e adaptar experiências passadas – chamadas casos – armazenadas em uma base de casos. Um novo problema é resolvido com base na adaptação de soluções de problemas similares já conhecidas.

Ainda pode-se dizer que são sistemas que usam o conhecimento representado explicitamente para resolver problemas. Manipulam o conhecimento e informações de maneira inteligente e tem por função auxiliar na resolução de problemas que requerem quantidade considerável de conhecimento humano. (REZENDE, 2003).

Dessa maneira pode-se afirmar que o RBC está focado na solução de problemas e aprendizado com base de experiências passadas. Um problema é resolvido com a reutilização da solução de um problema anterior parecido com o problema atual. Pode funcionar como modelo cognitivo para entender aspectos do comportamento e pensamento dos seres humanos. Possui um conhecimento específico, focado em exemplos concretos de casos, totalmente orientado a metas e soluções, diferentemente de outras técnicas de inteligência artificial. (WANGENHEIM; WANGENHEIM, 2003).

Um sistema de RBC deve ser capaz de questionar o usuário, usando linguagem de fácil entendimento, desenvolver linha de raciocínio a partir dessas informações e do conhecimento nele inserido, explicar seu raciocínio devendo ser capaz de interpretar o processo e apresentá-lo de forma compreensível. (REZENDE, 2003).

O padrão mais aceito para o processo de RBC é o Ciclo de RBC, que é constituído por um ciclo de raciocínio contínuo, composto pelas tarefas de recuperar, reutilizar, revisar e reter um caso. Conforme o problema, a base de dados é pesquisada para buscar tarefas anteriormente resolvidas, cuja descrição seja simular a atual. (WANGENHEIM; WANGENHEIM, 2003).

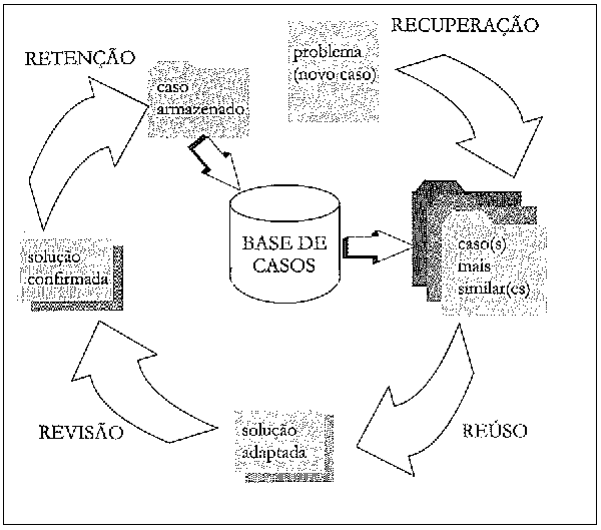

Figura 1 - Ciclo do Raciocínio Baseado em Casos (WANGENHEIM; WANGENHEIM, 2003, p.15)

2.2 Tecnologias Web

Com a disseminação das aplicações web, muitas plataformas ganharam popularidade, como foi o caso do PHP e do ASP. No entanto, não demorou muito para que o Java também suportasse o desenvolvimento web, com o uso de modelos como o MVC (*Model-View-Controller*), que possibilita a separação da aplicação web em diversas tarefas e camadas (CORDEIRO, 2012).

Segundo Gonçalves (2007, p.385)

> MVC é um conceito (paradigma) de desenvolvimento e design que tenta separar uma aplicação em três partes distintas. Uma parte, a Model, está relacionada ao trabalho atual que a aplicação administra, outra parte, a View, está relacionada a exibir os dados ou informações dessa uma aplicação e a terceira parte, Controller, em coordenar os dois anteriores exibindo a interface correta ou executando algum trabalho que a aplicação precisa completar.

O padrão MVC é divido nas camadas de Modelo (*Model*), Visão (*View*) e Controlador (*Controller*). A Visão representa a camada de interação com o usuário, as telas, que podem ser formadas por JSP, HTML, Javascript e CSS. O Modelo é a camada onde ficam as regras de negócio, processamentos e integrações. Controlador é o responsável por receber as informações da visão, montar os objetos e transmiti-las ao modelo, que ao completar o processo retorna ao usuário o que foi executado (CORDEIRO, 2012).

Para facilitar o desenvolvimento da Visão, surgiram *frameworks* que deixaram mais simples e produtiva a implementação dessa camada, sendo que um desses é o JSF (*Java Server Faces*).

O uso do Java Server Faces tornou o ambiente de desenvolvimento web mais fácil, através do uso de componentes de interface com o usuário, possibilitando a conexão desses componentes a objetos de negócios de forma simplificada (GONÇALVES, 2008).

Segundo Gomes (2008, p.11)

> O JSF (*Java Server Faces*) é a tecnologia padrão do J2EE 1.4 (ou superior) para criar aplicações web. Ele herda das tecnologias JSP e Servelets e estende seus conceitos com um ciclo de vida e um conjunto de componentes e recursos sofisticados e focados no desenvolvimento RAD para web.

O *framework* JSF é a tecnologia responsável pela interação do sistema web com o usuário, fornecendo ferramentas para criar a interface visual e comunicar a camada de apresentação com a camada de negócios da aplicação. No entanto, a persistência com o banco de dados e outras conexões *back-end* estão fora do escopo do JSF (GONÇALVES, 2008).

Para resolver esse problema de persistência com o banco de dados é necessário utilizar outros *frameworks*, sendo que um muito

utilizado pelos desenvolvedores Java é o JPA (*Java Persistence API*).

O JPA é uma especificação criada para permitir o mapeamento objeto-relacional com bancos de dados relacionais (figura 2). Ao invés de salvar dados em tabelas, o próprio código do sistema carrega a persistência de classes com valores que se quer salvar. Assim os produtos que implementam JPA, transformam as requisições de consulta ou salvamento via classe, em comandos SQL que são enviados ao banco de dados (GOMES, 2008).

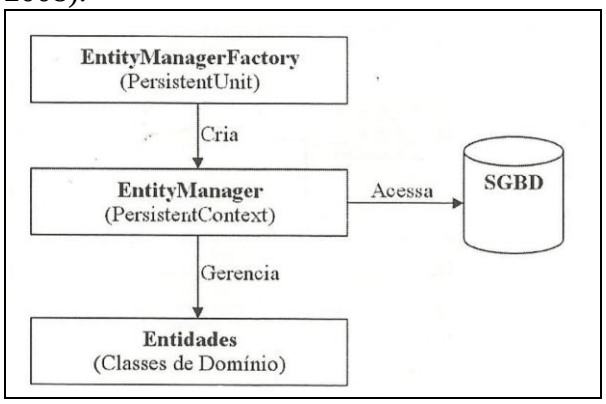

Figura 2 – Arquitetura do JPA (GOMES, 2008, p.47)

Para realizar esse processo de mapeamento objeto relacional com JPA, são utilizadas anotações nas classes que em conjunto com a unidade de persistência, permitem a comunicação das classes da aplicação web com o banco de dados. As principais anotações usadas nas classes são:

- *@Entity* – Utilizada para identificar que a classe é uma entidade.
- *@Table* – Utilizada para identificar a estrutura da tabela a ser criada no banco de dados.
- *@Id* – Indica que o atributo é chave primária.
- *@GeneratedValue* – Indica como será gerado os valores do atributo.
- *@Column* – Indica como será a estrutura do campo no banco de dados.
- *@Temporal* – Utilizada para formatar atributos de tipos especiais, como por exemplo, campos do tipo Date.
- *@OneToOne, @OneToMany, @ManyToMany, @ManyToOne* - Indicam o relacionamento de uma entidade (classe) com outra entidade.

2.3 Trabalhos correlatos

Barbosa (2012) propôs em seu trabalho de conclusão de curso (TCC) uma aplicação web de *help desk* com o uso de RBC para auxiliar no processo de suporte técnico aos clientes que utilizam o sistema da empresa EduSoft. O principal objetivo da ferramenta desenvolvida foi garantir um mecanismo de suporte à decisão mais ágil e preciso. Neste trabalho foi utilizada a linguagem de programação PHP e o banco de dados SqlServer.

Cavalari e Costa (2005) apresentaram em seu artigo o desenvolvimento um sistema de *help desk* para a prefeitura municipal de Lavras utilizando Raciocínio Baseado em Casos e Regras. A ferramenta implementada visa melhorar o suporte aos usuários que trabalham com a infraestrutura tecnológica da prefeitura e possibilitar o gerenciamento do inventário de hardware e software do local. A linguagem de programação utilizada foi PHP e o banco de dados MySql.

Fernandes**,** Guckert e Moreira (2010) aplicaram técnicas de RBC no desenvolvimento de uma ferramenta de *help desk* com ênfase na sua FAQ, a fim de auxiliar no suporte a decisão de problemas relatados por usuários que utilizam o sistema ECG. Utilizando liguagem de programação PHP à ferramenta foi implementada uso de usuários do sistema e técnicos do suporte técnico.

3. Estudo de caso

Com base nos conceitos de raciocínio baseado em casos e Java para web, foi desenvolvido um protótipo de sistema de help desk para auxilio no suporte a decisão de uma equipe de suporte técnico.

A implementação do sistema web foi constituída pelo uso da linguagem de

programação Java, com o auxilio dos *frameworks* para desenvolvimento web JSF e JPA em conjunto do padrão MVC. O servidor web utilizado foi o Glassfish e o banco de dados PostgreSql.

3.1 Apresentação do sistema

A aplicação foi projetada para ser usada no âmbito do suporte a decisão de problemas relacionados aos módulos de um sistema de automação comercial, listando aos usuários do sistema as possíveis soluções para os problemas pesquisados. Uma característica desse sistema é que ele não trabalha com o tratamento de informações específicas, possibilitando aos usuários a criação de uma estrutura genérica para a busca de soluções.

Para que o sistema tenha um ótimo desempenho e um alto percentual de acerto nas soluções encontradas para os problemas pesquisados, é muito importante que a estrutura dos campos seja elaborada por um usuário especialista, que por entender do domínio do sistema de automação comercial, sabe quais são as questões mais relevantes para a construção do layout de descrição do problema de cada módulo.

Com a configuração do ambiente de busca concluída pelo especialista, o usuário técnico e cliente podem utilizar o sistema para a busca de possíveis soluções para um novo problema que possa ter sido encontrado em algum módulo do sistema de automação comercial.

Após o procedimento de busca ser concluído, o caso com maior grau de similaridade pode ser aplicado como solução para o problema atual, permitindo ao usuário armazenar esse novo caso. Sendo que esse caso poderá ser usado futuramente para auxiliar na solução de outros problemas. Posteriormente o caso salvo na base de dados poder ser revisado e revalidado.

Para facilitar o processo de busca de casos similares foi desenvolvido um dicionário de dados. Dicionário que foi implementado orientado ao uso de palavras chaves com suas respectivas similaridades. Por exemplo, se na descrição de um problema, no campo referente à descrição do módulo, o usuário inserir a palavra “NFe”, ao ser executado o procedimento de busca, o sistema irá verificar no dicionário de dados se existe alguma palavra chave ou mesmo derivações desta palavra, que no caso de “NFe” podem ser “nota fiscal”, “nota fiscal eletrônica” ou “NF”. Então, com base nessa coleção de palavras o sistema irá buscar por casos que contenham em sua descrição essas palavras chaves. Através disso o processo de busca será mais robusto e preciso para o encontro do caso mais similar.

A figura 3 exibe o processo de funcionamento geral do sistema.

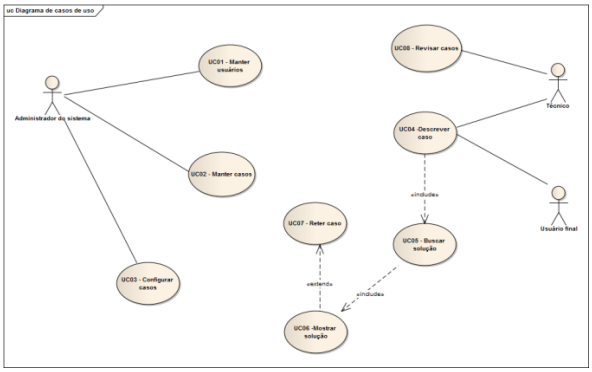

Figura 3 – Diagrama de casos de uso do sistema web de RBC

Para que o processo de busca e aquisição de conhecimento funcionasse corretamente no contexto do sistema web, foram utilizadas as seguintes técnicas e métodos do RBC:

- Para a representação dos casos foi implementada o método de atributo por valor.
- A medida de similaridade adotada foi a simétrica, onde se o caso A é igual a B, então B é igual a A.
- Para a similaridade global foi usada à técnica do *nearest neighbour* (vizinho mais próximo) ponderado.
- A medida de similaridade local para atributos numéricos adotada foi a tipo escalar.
- Foi utilizada a contagem de palavras para a medida da

similaridade local para atributos de tipo texto.

- A recuperação dos casos foi implementada através do método de recuperação sequencial.
- Para o processo de adaptação dos casos foi implementado o método de adaptação substitucional.
- A retenção dos casos foi desenvolvida com base no algoritmo de retenção das soluções por problemas descritos.

3.2 Estrutura e Persistência dos dados

No desenvolvimento do sistema web, inicialmente foram criadas as classes que formam a camada *Model* da aplicação. Nessas classes foram adicionados os atributos e as anotações derivadas do *framework* JPA, necessários para a comunicação dessas classes com o banco de dados. A estrutura da camada *Model* da aplicação foi constituída das classes indicadas na figura 4.

Após a criação das classes, o passo seguinte foi adicionar uma unidade de persistência ao sistema. Nessa unidade os seguintes parâmetros precisaram ser configurados:

1. Nome da Unidade de persistência – Adição de um nome para a unidade criada.
2. Provedor de persistência – Indicação do *framework* JPA necessário para prover a persistência do banco de dados.
3. Fonte de dados – Adição do serviço de fonte de dados do PostgreSql, que contém as informações referentes ao banco criado para o sistema web.
4. Classes da entidade – Inseridas todas as classes (figura 4) responsáveis pela criação da estrutura do banco de dados.

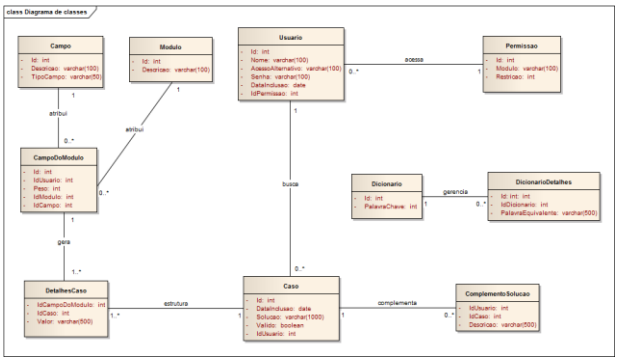

Figura 4 – Diagrama de classes do sistema web de RBC

Em seguida foi criada a classe responsável pela persistência dos dados da aplicação web com banco de dados. Essa classe foi denominada de JpaUtil. Nela foram inseridos os parâmetros *EntityManagerFactory* e *EntityManger*, que são responsáveis em conjunto com a unidade de persistência pela montagem da estrutura do banco de dados e sua respectiva conexão. Além disso, nessa classe foram adicionados os métodos responsáveis pela introdução, alteração, exclusão e consulta de registros no banco de dados.

Para finalizar o processo de criação da persistência do sistema web com o banco de dados, foi criada a classe Listener, que é responsável pela chamada da classe que contém a inicialização do serviço de conexão do banco de dados, que no caso do sistema web é a classe JpaUtil. Dessa maneira ao iniciar o sistema, o método de inicialização da classe Listener é chamado e consequentemente a estrutura e a conexão do banco de dados são criadas.

3.3 O uso do MVC no desenvolvimento do sistema

O padrão de desenvolvimento web MVC foi adotado na implementação da aplicação web para possibilitar a separação interna do sistema em camadas, resultando na organização e facilidade de manutenção do código escrito, além de possibilitar a utilização de muitos recursos provenientes dos *frameworks* JSF e JPA específicos para o ambiente web com MVC.

Na camada *Model* da aplicação web, foram incluídas as classes que fazem o controle e manejo dos dados dos objetos. Como apresentado no tópico anterior, essas classes possuem anotações que são usadas para a comunicação com o banco de dados. Essa camada ficou responsável pelas transações de dados dos objetos da aplicação web com o as tabelas do banco de dados.

Na camada *Controller* da aplicação, foram criadas as classes responsáveis pela comunicação entre a camada *View* e *Model* do sistema web. Através das anotações JSF *@ManagedBean* e *@SessionScoped* essas classes tem seus métodos e atributos visíveis na camada *View*. Assim é possível que esses métodos e atributos da classe controladora sejam adicionados aos componentes de tela da aplicação, para que sejam executados nos possíveis eventos que o usuário irá realizar durante o uso do sistema.

Essas classes controladoras têm métodos e atributos para validações que são utilizados para impedir o usuário de realizar determinados procedimentos na camada *View*, métodos de execução dos algoritmos de RBC necessários para busca das soluções de problemas informados na aplicação web e comunicação com os objetos e métodos de persistência com o banco de dados na camada *Model*. A camada *Controller* da aplicação é constituída de classes controladoras para todas as entidades indicadas na figura 4.

Por fim na camada *View*, foram criados os arquivos de interação com o usuário do sistema. Com o uso do framework JSF foi possível realizar um desenvolvimento web simples, voltado ao uso de componentes e de fácil ligação com a camada *Controller*. O JSF proporcionou a criação de telas de interação com o usuário, através do uso de caixas de texto, botões, painéis, listas, *grids* e outros componentes espalhados em outros arquivos pela camada *View*. Um fator importante do JSF é a facilidade de comunicação dos componentes com as classes de controle. Comunicação que é feita através da propriedade *value* dos componentes para atributos e comando *action* para os métodos. Uma característica importante é que para a execução desses métodos e comunicação com atributos, é necessário que a declaração seja feita entre os caracteres *#{...}*, como indica a figura 5.

```
<h:panelGrid  columns="2" id="grid2">
    <h:outputText value="Descreva a solução do problema :"/>
    <p:inputTextarea rows="10" cols="50"
                     value="#{casoBean.caso.solucao}"
                     />
</h:panelGrid>
<p:outputPanel layout="block"
               style="margin: auto; width:40%">
    <p:commandButton value="Salvar" action="#{casoBean.salvar}"
                     update="salvo"/>
    <p:commandButton value="Cancelar" ajax="false"
                     immediate="true"
                     action="#{casoBean.cancelar}"/>
    <p:dialog id="salvo" header="Salvo"
              widgetVar="ss" closable="false"
              modal="true" visible="#{casoBean.salvo}" >
        <h:outputText value="Campo salvo com sucesso"/><br/>
        <p:commandButton value="Fechar" ajax="false"
                         action="#{casoBean.cancelar}"/>
    </p:dialog>
</p:outputPanel>
```

Figura 5 – Representação do uso de JSF na camada View

3.4 Resultados

Para validar o sistema web de suporte a decisão foram realizados testes com alguns técnicos da equipe de suporte técnico. Nos primeiros testes ficou evidente que as soluções pesquisadas na base de dados foram em geral pouco similares com os problemas descritos. Inicialmente a base continha cerca de trinta casos cadastrados, referentes a vários módulos do sistema de automação comercial, dessa maneira deixando claro que o sistema possuía um baixo nível de conhecimento armazenado. Conclui-se nessa primeira etapa que devido à base de dados estar com poucos casos cadastrados e por não ter um usuário especialista configurando a tela de descrição de problemas por módulo, a margem de acerto estava muito baixa. Na tabela 1 estão exibidos os resultados iniciais das buscas por soluções para problemas por módulo.

Módulo	Nº de pesquisas	% médio de acerto
NFE	20	30
SPED Fiscal	17	15
EFD Contribuições	19	22
Relatórios	14	18
Compras	10	20

Tabela 1 – Resultados iniciais

No entanto conforme novos casos foram sendo adicionados à base de dados e um usuário especialista foi colocado para configurar a tela de cadastro de problemas por módulo, a margem de acerto começou a subir gradativamente. A base de dados no encerramento dos testes estava com cerca de duzentos casos inseridos. Os módulos que possuíam um número maior de casos resolvidos foram os módulos de nota fiscal eletrônica, gerador de relatórios, Sped Fiscal e EFD contribuições. Na tabela 2 estão exibidos os resultados finais das buscas por soluções para problemas por módulo.

Módulo	Nº de pesquisas	% médio de acerto
NFE	50	70
SPED Fiscal	38	78
EFD Contribuições	43	65
Relatórios	44	73
Compras	27	55

Tabela 2 – Resultados finais

Por fim concluímos que quanto maior o número de casos armazenados e com o auxílio de um usuário especialista monitorando o uso do sistema web, o percentual de acerto na indicação da solução para um novo problema será cada vez maior. No entanto, para problemas que possuem baixo grau de similaridade com os casos contidos na base, haverá a necessidade de que o técnico que concluir a tarefa insira uma descrição completa e correta para a solução do problema resolvido, fazendo com que futuramente o sistema possa utilizar esse caso como base de conhecimento para a solução de outros problemas. A figura 6 exibe a tela de pesquisa de soluções para um novo problema descrito.

Figura 6 – Tela de pesquisa por soluções do sistema web

4. Conclusão

Com o desenvolvimento desse sistema web foi possível perceber que a técnica de raciocínio baseado em casos, em conjunto com um dicionário de palavras, auxiliaram no suporte a decisão para equipes de suporte técnico, resultando em um percentual maior de soluções similares e corretas para os problemas descritos.

As tecnologias web utilizadas possibilitaram a implementação de um sistema simples, moderno e apropriado para o suporte à decisão utilizando as técnicas de RBC, através do uso da linguagem de programação Java com os *frameworks* JSF e JPA em conjunto do padrão de desenvolvimento web MVC.

No entanto é importante frisar que o sistema web não substitui por completo a decisão do técnico, mas o auxilia na tomada de uma decisão para que evite perda de tempo na busca de possíveis soluções e, como consequência, cause danos econômicos e estruturais às informações do cliente.

Como sugestão para trabalhos futuros pode se aprimorar o desenvolvimento do sistema web utilizando outras técnicas de inteligência artificial; implementar um dicionário de dados mais completo, com adjetivos, pronomes, verbos; e estender a aplicação para outras áreas, como por exemplo, saúde humana, jurídica e meteorologia.

Referências

BARBOSA, Rodrigo Rafael. **Aplicação de uma FAQ baseada em técnicas de RBC no suporte a decisão aos clientes da empresa EDUSOFT Tecnologia LTDA**. Blumenau, 2012. Disponível em: <http://campeche.inf.furb.br/tccs/2012-I/TCC2012-1-16-VF-RodrigoRBarbosa.pdf>. Acesso em 02 Dez. 2012.

CAVALARI, Gabriel. O. T.; COSTA, Heitor. A. X. **Modelagem e Desenvolvimento de um Sistema Help-Desk para a Prefeitura Municipal de Lavras - MG**. Revista Facela. Curitiba, 2005. Disponível em: <http://revistas.facecla.com.br/index.php/reinfo/article/viewFile/158/50>. Acesso em: 02 Dez. 2012.

CORDEIRO, Gilliard. **Aplicações Java para web com JSF e JPA.** São Paulo: Casa do Código, 2012.

FERNANDES, Anita Maria da Rocha; GUCKERT, Richard Marthendal; MOREIRA, Daniela Souza; **Aplicação de uma FAQ Baseada em RBC para Suporte a Usuários de um Sistema Web**. Criciúma, 2010. Disponível em: <http://periodicos.unesc.net/index.php/sulcomp/article/download/281/289.pdf>. Acesso em: 02 Dez. 2012.

GOMES, Yuri Marx P. **Java na Web com JSF, Spring, Hibernate e Netbeans 6.** Rio de Janeiro: Ciência Moderna, 2008.

GONÇALVES, Edson. **Desenvolvendo Aplicações Web com JSP, Servlets, JavaServer Faces, Hibernate, EJB 3 Persistence e Ajax**. Rio de Janeiro: Ciência Moderna, 2007.

GONÇALVES, Edson. **Dominando Java Server Faces e Facelets utilizando Spring 2.5, Hibernate e JPA**. Rio de Janeiro: Ciência Moderna, 2008.

REZENDE, Solange Oliveira. **Sistemas Inteligentes: Fundamentos e Aplicações.** Barueri: Manole, 2003.

WANGENHEIM, Christiane Gresse Von, WANGENHEIM, Aldo Von. **Raciocínio Baseado em Casos.** Barueri: Manole, 2003.