HELTON MACHADO KRAUS

FERRAMENTA PARA DESENVOLVIMENTO DE SISTEMAS DE RACIOCÍNIO BASEADO EM CASOS

São José (SC), dezembro de 2009

UNIVERSIDADE DO VALE DO ITAJAÍ

CURSO DE MESTRADO ACADÊMICO EM COMPUTAÇÃO APLICADA

FERRAMENTA PARA DESENVOLVIMENTO DE SISTEMAS DE RACIOCÍNIO BASEADO EM CASOS

por

Helton Machado Kraus

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Computação Aplicada.
Orientadora: Anita Maria da Rocha Fernandes, Dra. Eng.

São José (SC), dezembro de 2009

FOLHA DE APROVAÇÃO

"FERRAMENTA PARA DESENVOLVIMENTO DE SISTEMAS DE RACIOCÍNIO BASEADO EM CASOS".

Helton Machado Kraus

'Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Computação Aplicada, Área de Concentração Computação Aplicada e aprovada em sua forma final pelo Programa de Mestrado em Computação Aplicada da Universidade do Vale do Itajaí.'

Prof. Anita Maria da Rocha Fernandes, Dra.
Orientadora

Prof. Cesar Albenes Zeferino, Dr.
Coordenador do Programa Mestrado em Computação Aplicada

Apresentado perante a Banca Examinadora composta pelos Professores:

Prof. Anita Maria da Rocha Fernandes, Dra. (UNIVALI)
Presidente

Prof. Rudimar Luis Scaranto Dazzi, Dr. (UNIVALI)
Membro

Prof. Raimundo Celeste Ghizoni Teive, Dr. (UNIVALI)
Membro

Profª. Silvia Modesto Nassar, Drª. (UFSC/PPCComp)
Membro

Prof. Fernando Mendes de Azevedo, Dr. (UFSC/PGEEL)
Membro

São José (SC), 11 de dezembro de 2009.

FERRAMENTA PARA DESENVOLVIMENTO DE SISTEMAS DE RACIOCÍNIO BASEADO EM CASOS

Helton Machado Kraus

Dezembro / 2009

Orientadora: Anita Maria da Rocha Fernandes, Dra. Eng.

Área de Concentração: Computação Aplicada

Linha de Pesquisa: Inteligência Aplicada

Palavras-chave: Inteligência Artificial. Raciocínio Baseado em Casos. Ferramenta para Desenvolvimento.

Número de páginas: 146

RESUMO

O Raciocínio Baseado em Casos (RBC) estabeleceu-se nos últimos anos como uma das tecnologias mais populares para o desenvolvimento de Sistemas Baseados em Conhecimento. A técnica de RBC busca a solução para uma situação atual através da recuperação e adaptação de soluções passadas semelhantes, dentro de um mesmo domínio do problema. Neste contexto, o RBC pode funcionar inclusive como um modelo cognitivo para se entender alguns aspectos do pensamento e comportamento humano, além de ser uma tecnologia simples de se usar para construir sistemas computacionais inteligentes e resolver problemas reais em diversas áreas. Para o desenvolvimento deste tipo de sistema existem ferramentas que fornecem funções para elaboração e testes, porém obrigam os usuários a terem pleno domínio da técnica de RBC. A falta de uma interface amigável para definição da representação do conhecimento e das métricas de similaridade é outro ponto que as ferramentas atuais não disponibilizam. O presente trabalho apresenta o desenvolvimento de uma ferramenta para o desenvolvimento de sistemas RBC com foco no ensino da técnica, que possibilite o uso em turmas de graduação da disciplina de Inteligência Artificial. Para ensino, a ferramenta seguirá como estratégia a Aprendizagem Baseada em Problemas, que é uma abordagem que utiliza problemas do mundo real, estudos de caso hipotéticos com resultados concretos e convergentes para que os aprendizes assimilem o conteúdo planejado e desenvolvam a habilidade de pensar criticamente. A ferramenta utiliza como base o framework jColibri 2, que é Open Source e modela a estrutura dos sistemas RBC, disponibilizando um ambiente onde os usuários possam entender o funcionamento da técnica de RBC, auxiliados por um assistente e por ajudas contextualizadas, como também elaborar novos sistemas interagindo com a ferramenta. A ferramenta foi validada com alunos do Curso de Ciência da Computação, na Disciplina de Inteligência Artificial.

TOOL FOR DEVELOPMENT OF CASE-BASED REASONING SYSTEMS

Helton Machado Kraus

December / 2009

Supervisor: Anita Maria da Rocha Fernandes, Dra. Eng.

Area of Concentration: Applied Computer Science

Line of Research: Artificial Intelligence

Key words: Artificial Intelligence. Case-based Reasoning. Tool for Development.

Number of pages: 146

ABSTRACT

Case-Based Reasoning (CBR) has established itself in recent years as one of the most popular technologies for the development of Knowledge Based Systems. CBR systems look for a solution to the current situation by restoring and adapting past solutions to similar situations, within the same area of the problem. In this context, CBR can even work as a cognitive model for understanding aspects of human thought and behavior, besides being a technology that is simple to use for building intelligent computer systems and solving real problems in several areas. For the development of this type of system, there are tools that provide system functions for preparation and testing, but these require users to have a good knowledge of the CBR technology. The lack of a friendly interface for defining the representation of knowledge and similarity metrics is another aspect currently not provided by the tools. This work aims to develop a tool for the development of CBR systems, focusing on teaching the techniques involved, enabling its use in graduate classes in the discipline of Artificial Intelligence. The teaching tool will follow the strategy of Problem-Based Learning, which is an approach that uses real world problems; hypothetical case studies with concrete and converging results, enabling learners to assimilate the planned content and develop the ability to think critically. The tool uses the jColibri 2 framework, which is Open Source, and models the structure of CBR systems, providing an environment where users can understand the functioning of the technique of CBR, aided by an assistant and contextualized help, but also develop new systems that interact with the tool. The tool was validated with students of Computer Science of the Department of Artificial Intelligence.

LISTA DE ILUSTRAÇÕES

LISTA DE EQUAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ABP	Aprendizagem Baseada em Problemas
API	Application Programming Interfaces
AWT	Abstract Window Toolkit
CBR	Case Based Reasoning
ECCER	European Conference on Case-Based Reasoning
GAIA	Group of Artificial Intelligence Applications
GDA	Algoritmo Gradiente Descendente
IA	Inteligência Artificial
IDE	Integrated Development Environment
ICCBR	International Conference on Case-Based Reasoning
IDF	Inverse Document Frequency
IHC	Interação Humano-Computador
INRIA	Instituto Nacional de Pesquisa em Automação e Informática da França
JDBC	Java Database Connectivity
KADS	Knowledge-Based System Development
LISP	List Processing Language
LOOM	Ontology Markup Language
MOP	Memory Organization Packets
PBL	Problem Based Learning
RBC	Raciocínio Baseado em Casos
RDN	Redes Discriminatórias Redundantes
SQL	Structured Query Language
SWT	Standard Widget Toolkit
TF	Term Frequency
UFSC	Universidade Federal de Santa Catarina
UML	Unified Modeling Language
UNIVALI	Universidade do Vale do Itajaí
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

A Inteligência Artificial (IA) é uma área de pesquisa da Ciência da Computação que estuda a representação de conhecimento e a capacidade das máquinas de pensar e aprender a resolver problemas de maneira similar a mente humana (FERNANDES, 2003).

A IA tem como objetivo modelar o conhecimento humano através de esquemas computacionais e neste contexto existem diferentes técnicas sendo empregadas, tais como a técnica de Raciocínio Baseado em Casos (RBC). Sistemas de RBC permitem a extração, organização e reuso do conhecimento utilizado para tomada de decisões no passado, tornando explícitos os métodos utilizados e permitindo o seu aperfeiçoamento. Cada caso contém tipicamente uma descrição do problema mais a solução e/ou resultado. Pode possuir um conjunto de sucessos e insucessos de casos resolvidos (AAMODT & PLAZA, 1994). Esta técnica surgiu de um modelo de armazenamento de informações da memória humana, sendo utilizada como uma técnica para resolução de novos problemas baseados em soluções resolvidas anteriormente, dentro de um mesmo domínio do problema (WATSON, 1997).

Os sistemas RBC tiveram sua inspiração inicial em 1977, quando Schank e Abelson no trabalho conhecido como Memória Dinâmica, propuseram que o conhecimento geral das pessoas sobre determinado assunto está armazenado em *scripts* ou roteiros. Outras linhas de pesquisa contribuíram para o desenvolvimento de sistemas RBC, sendo que o primeiro sistema realmente desenvolvido para funcionar como um sistema baseado em casos foi denominado de CYRUS, desenvolvido por Janet Kolodner, baseado no modelo de memória dinâmica de Schank (AAMODT & PLAZA, 1994).

Kolodner (1993) estabelece que um sistema de raciocínio baseado em casos resolve problemas mediante a reutilização do conhecimento e experiências recuperadas de uma situação-problema prévia mais similar, a partir de uma base de casos. A recuperação dos dados é realizada em memória, verificando se existem casos semelhantes com as características atuais do problema, podendo encontrar um ou mais casos e adaptá-los de alguma maneira, para que se ajustem ao problema atual, criando um novo caso para uso futuro.

Esta técnica é composta basicamente por quatro elementos: a) **Representação do Conhecimento** (representado principalmente em forma de casos, que descrevem experiências

concretas); b) **Medida de Similaridade** (define como será calculada a similaridade entre a situação atual e um determinado caso na base de casos); c) **Adaptação** (mecanismos para adaptar os casos recuperados completamente, para verificar se satisfazem às características da situação presente); e d) **Aprendizado** (para que um sistema se mantenha atualizado e evolua continuamente, sendo capaz de lembrar dessa situação no futuro como mais um novo caso) (WANGENHEIM & WANGEHEIM, 2003; FERNANDES, 2003).

Os sistemas de RBC possuem muitas características relevantes para sua utilização, porém um problema importante que deve ser tratado é o armazenamento e o processamento necessário para recuperação dos casos na base de casos. Com isso desenvolvedores deste tipo de sistema necessitam analisar as diferentes representações de conhecimento e métricas de similaridade para definir qual a mais adequada para cada tipo de domínio de aplicação. A má definição destes dois elementos pode trazer problemas na eficiência dos sistemas de RBC.

Para a criação e aplicação deste tipo de sistema, os desenvolvedores devem optar por criar sua própria ferramenta, programando toda a lógica do sistema, ou utilizando uma das ferramentas de desenvolvimento existentes.

Os diferentes tipos de ferramentas voltadas ao desenvolvimento de aplicações de RBC podem ser classificadas em três grupos de acordo com Jaczynski e Trousse (1998): ferramentas denominadas de CBR *shells*, Interfaces de Programas Aplicativos RBC, e *frameworks* para RBC.

As ferramentas denominadas CBR *shells* permitem a um usuário, em geral não programador, a geração de aplicações de RBC por meio de uma interface gráfica sofisticada. Nesta interface, os parâmetros necessários ao desenvolvimento da aplicação podem ser definidos de modo interativo pelo usuário, como por exemplo especificar os descritores dos casos relativos ao domínio de conhecimento, bem como o vetor de pesos usados para o cálculo da medida de similaridade usada para recuperar casos. Entretanto, usualmente, estas ferramentas não permitem modificações ou integração de novos componentes (JACZYNSKI & TROUSSE, 1998). Um exemplo desta classificação é a ferramenta computacional CBR-Works (CBR WORKS, 2008), que possibilita definir as características do sistema de RBC através de sua interface gráfica.

As Interfaces de Programas Aplicativos (*Application Programming Interfaces* ou APIs) fornecem um conjunto de funções para gerenciar os algoritmos de RBC, e são projetadas para serem usadas por programadores, permitindo uma customização limitada por meio da respectiva

linguagem de programação, visando adicionar, por exemplo, novas medidas de similaridade ou técnicas de adaptação. Entretanto, o propósito destas interfaces não é oferecer componentes de software genéricos ou de código aberto, mas somente customizar as entradas e saídas do sistema (JACZYNSKI & TROUSSE, 1998). A ferramenta CBR*Tools (CBR TOOLS, 2008) é um exemplo desta classificação.

Por sua vez, um *framework* (ou arcabouço) para métodos de RBC, segundo Fayad, Schimidt e Ralph (1999), consiste de uma aplicação de software semicompleta e reutilizável que pode ser especializada para produzir aplicações customizadas. Um exemplo desta classificação é o *framework* jColibri 2.1 (GAIA, 2008).

As soluções para desenvolvimento de RBC fornecem funções para elaboração de sistemas, porém obrigam na maioria dos casos os usuários a terem pleno domínio da técnica de RBC, conhecimento pelos quais alunos de cursos de graduação ou programadores em geral possuem de forma limitada, por não aplicarem a técnica em situações reais. Outro agravante é a falta de uma interface amigável para definição da representação do conhecimento e das métricas de similaridade.

Observando a complexidade envolvida no desenvolvimento dos sistemas de RBC e a falta de soluções apropriadas para o ensino e desenvolvimento, pretende-se realizar o levantamento do estado da arte deste tipo de sistema. Também se pretende realizar uma análise das soluções similares e então propor uma ferramenta para elaboração de sistemas de RBC.

A ferramenta é um *Shell* para desenvolvimento de sistemas RBC, que possibilita aos usuários utilizarem, ou estenderem, as definições elaboradas para outras aplicações externas. O desenvolvimento leva em consideração a ergonomia e usabilidade de sistemas para criar um ambiente de ensino e desenvolvimento de sistemas de Raciocínio Baseado em Casos, possibilitando o uso na academia para servir de auxílio aos educadores e gerar facilidade aos alunos.

1.1 PROBLEMA DE PESQUISA

Para o desenvolvimento de sistemas de RBC as soluções existentes fornecem funções para elaboração do sistema, porém obrigam os usuários a terem pleno domínio da técnica de RBC. A falta de uma interface amigável e consistente para definição da representação do conhecimento e das métricas de similaridade é outro ponto que estas soluções não disponibilizam.

Outra característica que deve ser levada em consideração na elaboração de sistemas de RBC é o uso de diferentes tecnologias, desde a forma como são armazenadas as informações (sistemas de arquivos ou banco de dados), algoritmos de recuperação e indexação das informações, métodos de adaptação, até as interfaces para apresentação das informações aos usuários.

Com uma análise prévia realizada em trabalhos acadêmicos anteriores, foi possível constatar a falta de auxílio na definição das características do sistema de RBC, onde não é apresentada uma ajuda consistente. Também foi constatada uma dificuldade em utilizar as ferramentas existentes no meio acadêmico como uma forma de ensino deste tipo de sistema.

Sendo assim, este trabalho analisa as soluções existentes para o desenvolvimento de sistemas de RBC, apresentando as principais características, como também a usabilidade, licença de uso, formas de armazenamento da base de casos, entre outras características.

Previamente sabe-se que existem soluções comerciais, que necessitam da aquisição do produto para uso, como também soluções distribuídas como software livre, que possuem seu código fonte disponível. Com isso tem-se como problema de pesquisa a análise das soluções de desenvolvimento de RBC existentes, levantando suas principais características e deficiências, para propor uma ferramenta para elaboração deste tipo de sistema.

1.1.1 SOLUÇÃO PROPOSTA

Como tema de pesquisa, busca-se levantar o funcionamento dos sistemas de RBC e a usabilidade das ferramentas existentes a fim de propor uma ferramenta para elaboração de sistemas RBC, que possibilite o desenvolvimento desse tipo de sistema, tanto para soluções acadêmicas como comerciais, como também para o ensino da técnica.

Para isso, a ferramenta possibilita desenvolvedores, com pouca experiência na área de Inteligência Artificial, desenvolver um sistema RBC com facilidade, auxiliado por um assistente que guia o usuário aplicando uma estratégia pedagógica para o ensino da técnica.

A solução proposta abrange:

- Definição do problema: auxílio na definição das características do sistema RBC, modelagem dos dados e definição dos tipos de informação;
- Armazenamento das informações: forma como as informações serão armazenadas, seja em sistemas de arquivos ou banco de dados, gerando ou recuperando de forma automatizada as estruturas de armazenamento;
- Definição das características do sistema RBC: estabelecimento dos métodos de recuperação das informações e métricas de similaridade que serão utilizados;
- Critérios para avaliação dos casos e armazenamento de novas situações: adaptação e retenção de novos casos (retroalimentação do sistema); e
- Exportação da Aplicação: possibilidade de exportar as definições realizadas na ferramenta para serem utilizadas em aplicações específicas do usuário.

De acordo com estas características foi desenvolvida a ferramenta, baseada em parâmetros de ergonomia e usabilidade, com foco principal no ensino da técnica de RBC, aplicando a estratégia de Aprendizagem Baseada em Problemas, que visa a participação do aluno, o desenvolvimento de sua capacidade crítica e de auto-aprendizagem.

A ferramenta foi avaliada e validada em uma turma de Inteligência Artificial do curso de Graduação em Ciência da Computação, onde foi disponibilizada para alunos desenvolverem exercícios, que possibilitassem ser resolvidos com o uso da técnica de RBC, verificando desta forma sua aplicação no ensino da técnica.

1.1.2 DELIMITAÇÃO DE ESCOPO

O escopo contempla os objetivos gerais e específicos do trabalho, onde objetivou-se criar um ambiente de desenvolvimento de sistemas de Raciocínio Baseado em Casos, tendo como resultado uma ferramenta para o ensino da técnica junto a disciplina de Inteligência Artificial e também para o desenvolvimento de sistemas para uso em aplicações comerciais..

O desenvolvimento não levou em conta a avaliação o desempenho dos sistemas que venham a ser desenvolvidos, porém apresentará as informações necessárias para usuários, que não tenham conhecimento total da técnica, para elaborar este tipo de sistema.

A ferramenta possui funcionalidades para o usuário definir os requisitos pertinentes à elaboração de sistemas RBC, dispondo de métricas de similaridade (local e global) e métodos de adaptação que podem ser utilizados nas aplicações desenvolvidas, como também realizar testes para avaliar seu funcionamento. Além de desenvolver as aplicações, também pode ser utilizada para ensino da técnica, onde os professores elaboram exercícios para serem desenvolvidos pelos alunos.

1.1.3 JUSTIFICATIVA

Em trabalhos anteriores foi possível constatar a falta de soluções apropriadas para elaboração de sistemas de RBC, possuindo limitações quando a usabilidade (dificuldade na compreensão da arquitetura e nas definições que são necessárias) e também a falta de documentação e ajudas consistentes que facilite o entendimento e uso.

As áreas de aplicação da técnica de RBC são bastante abrangentes e envolvem diferentes tipos de problema (WANGENHEIM & WANGENHEIM, 2003). Alguns exemplos de problemas que podem ser resolvidos com o uso da técnica de RBC são sistemas de recomendação, diagnóstico, planejamento, apoio a decisão, etc. A falta de um ambiente de desenvolvimento apropriado para o uso acadêmico dificulta a utilização e o aprendizado da técnica.

Outro fator que comprova o uso e a importância da técnica é a quantidade significativa de artigos publicados em congressos da área de Inteligência Artificial, chegando a existir conferências internacionais sobre o tema, tais como o ECCER[1] (European Conference on Case-Based Reasoning ou Conferência Européia de Raciocínio Baseado em Casos) e o ICCBR[2] (International Conference on Case-Based Reasoning ou Conferência Internacional de Raciocínio Baseado em Casos). Ambos os eventos acontecem periodicamente, sendo que o ECCER está em sua nona edição, que aconteceu no ano de 2008, e o ICCBR está na décima oitava edição, que acontecerá em 2010, onde se pretende publicar parte do presente trabalho.

[1] http://www.wi2.uni-trier.de/eccbr08/index.php
[2] http://www.iccbr.org/iccbr10/

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

O objetivo deste trabalho é desenvolver uma ferramenta para construção de Sistemas de Raciocínio Baseado em Casos, com foco no ensino da técnica para o desenvolvimento de aplicações.

1.2.2 OBJETIVOS ESPECÍFICOS

- Analisar as formas de representação do conhecimento e métricas de similaridade dos sistemas RBC;
- Analisar soluções para desenvolvimento de sistemas de RBC;
- Analisar uma estratégia de ensino para aplicação em sistemas computacionais;
- Modelar a ferramenta, com base no *framework* jColibri 2.0;
- Desenvolver a ferramenta; e
- Validar a ferramenta junto à disciplina de Inteligência Artificial.

1.3 METODOLOGIA

Com a definição dos objetivos e o escopo do trabalho, nesta seção é apresentada uma síntese da metodologia da pesquisa e os procedimentos metodológicos utilizados durante a realização da dissertação.Metodologia da Pesquisa

A metodologia da pesquisa visa detalhar de forma minuciosa as ações desenvolvidas no método do trabalho de pesquisa. Desta forma uma pesquisa pode ser classificada de várias maneiras, sendo neste trabalho utilizada a pesquisa aplicada, onde geram-se conhecimentos para aplicação prática dirigida à solução do problema.

A natureza da pesquisa refe-se ao método qualitativo, que considera que há uma relação dinâmica entre o mundo real e o usuário, que não pode ser traduzido em números. Este método caracteriza-se por ser um estudo aprofundado de um sistema no ambiente onde ele está sendo usado, normalmente envolvendo pessoas ou sistemas, considerando a interpretação dos fenômenos e a atribuição de significados.

Dentro da pesquisa qualitativa este trabalho classifica-se como uma pesquisa observacional, que tem como objetivo observar o ambiente, mas não modificá-lo, ou seja, disponibiliza-se a ferramenta e observa-se quais as percepções obtidas pelos usuários quanto à facilidades e dificuldades obtidas com a utilização.

Do ponto de vista dos objetivos a pesquisa é observacional descritiva, pois busca descrever de forma objetiva e direta eventos e fatos de interesse, onde neste trabalho observou-se através de um questionário padronizado o desempenho da ferramenta durante o uso em uma turma do curso de Inteligência Artificial.Desta forma, teve-se como hipóteses de pesquisa:

- A ferramenta desenvolvida facilita a criação de sistemas de RBC.
- A ferramenta desenvolvida auxilia no ensino da técnica de RBC.

Estas hipóteses foram verificadas e validadas com a aplicação da ferramenta, e serão discutidas nos resultados do trabalho.

1.3.1 PROCEDIMENTOS METODOLÓGICOS

Os procedimentos metodológicos apresentam como o trabalho foi desenvolvido para atingir os objetivos propostos. Os procedimentos realizados seguiram etapas, onde cada uma resultou em artefatos, seja eles revisão bibliográfica, documentação ou parte da codificação da ferramenta. A seqüência de etapas realizadas no trabalho foram:

- Análise do funcionamento da técnica de RBC: Nesta etapa foi realizada a pesquisa bibliográfica dos conceitos que envolvem a técnica de RBC, buscando identificar as formas de representação do conhecimento, as métricas de similaridade, os métodos de adaptação e as demais definições pertinentes necessárias para elaboração deste tipo de sistema. Esta etapa resultou em uma documentação formal de parte dos requisitos do sistema, e também definiu parte do escopo geral da dissertação.
- Análise de soluções similares: Nesta etapa foram identificadas e analisadas as ferramentas de desenvolvimento de sistemas de RBC existentes, visando identificar suas principais características e deficiências, que serviram como base para elaboração da ferramenta proposta. O resultado desta etapa foi o comparativo entre as soluções, mostrando a diferença entre cada uma.

- Definição de uma estratégia de ensino: Foram analisadas abordagens pedagógicas para aplicação em sistemas computacionais, visando definir a estratégia de ensino e o conteúdo teórico que será utilizado na ferramenta para o ensino da técnica aos usuários que não possuem pleno domínio sobre RBC. Como resultado tem-se o detalhamento da abordagem e o material teórico que é disponibilizado,.
- Modelagem: os requisitos foram avaliados, desde as tecnologias utilizadas até as características finais, visando detalhar cada componente da ferramenta utilizando a linguagem UML (*Unified Modeling Language*). Este tipo de modelagem é composto por diagramas que descrevem as características de estrutura, comportamento e interação de sistemas computacionais.
- Desenvolvimento: Seguindo os requisitos e a modelagem gerada na etapa anterior foi desenvolvida a ferramenta, utilizando a linguagem de programação Java.
- Validação: Como forma de avaliar e validar a ferramenta, foram realizados testes junto à usuários, que são alunos da disciplina de Inteligência Artificial, do curso de Ciência da Computação. A validação visa verificar se a ferramenta auxilia no ensino da técnica de RBC.

1.4 ESTRUTURA DA DISSERTAÇÃO

O trabalho está organizado em 5 capítulos correlacionados. O Capítulo 1, Introdução, apresentou por meio de sua contextualização o tema proposto neste trabalho. Da mesma forma foram estabelecidos os resultados esperados por meio da definição de seus objetivos e apresentadas as limitações do trabalho permitindo uma visão clara do escopo proposto.

O Capítulo 2 descreve a fundamentação teórica do trabalho, apresentado a técnica de Raciocínio Baseado em Casos que serviu como base para o desenvolvimento de todo o trabalho e as ferramentas de desenvolvimento de RBC existentes. Este capítulo também apresenta uma breve descrição sobre métodos de ensino que podem ser aplicados à área computacional, descrevendo a Aprendizagem Baseada em Problemas, que servirá como abordagem para o ensino da técnica.

No Capítulo 3 é descrito o projeto e desenvolvimento da ferramenta, incluindo a modelagem e as atividades desenvolvidas, apresentando por fim as principais telas criadas.

No Capítulo 4 são apresentados os resultados obtidos com a aplicação da ferramenta no ensino da técnica, descrevendo a percepção dos alunos ao desenvolver aplicações utilizando RBC.

No Capítulo 5, tem-se as conclusões do trabalho, relacionando os objetivos identificados inicialmente com os resultados alcançados. São ainda propostas possibilidades de continuação da pesquisa desenvolvida a partir das experiências adquiridas com a execução do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Com a evolução das tecnologias, o computador a cada dia tem se tornado mais presente no cotidiano, realizando as mais diversificadas atividades, tais como tomadas de decisões e realizando tarefas automatizadas, algumas dessas envolvendo “raciocínio”. O computador aparece aqui como uma “extensão” da mente humana. Tornar possível a realização dessas tarefas por computadores é um dos objetivos da Inteligência Artificial. Segundo Fernandes (2003), Inteligência Artificial é a parte da Ciência da Computação voltada ao desenvolvimento de sistemas computacionais inteligentes com características associadas à inteligência do comportamento humano.

A popularidade das aplicações em Inteligência Artificial foi alta durante metade da década de 1980 e sofreu um declínio no começo dos anos 1990. A grande dificuldade enfrentada por estes sistemas foi a ineficiência de trabalhar com métodos puramente simbólicos para incorporar conhecimento. A nova geração de sistemas inteligentes está sendo construída com o desenvolvimento de sistemas híbridos que constituem um novo campo da Inteligência Artificial (THÉ, 2001).

A idéia principal dos sistemas de Raciocínio Baseado em Casos é resolver um novo problema relembrando uma situação anterior similar e, então, reutilizando informação e conhecimento daquela situação. Cada caso contém tipicamente uma descrição do problema mais a solução e/ou resultado. Também pode possuir um conjunto de sucessos e insucessos de casos resolvidos.

Desta forma a Seção 2.1 apresenta o funcionamento dos sistemas de Raciocínio Baseado em Casos, destacando as formas de representar e recuperar as informações no sistema e também as principais métricas de similaridade existentes que podem ser utilizadas.

Na Seção 2.2 apresenta-se o levantamento das ferramentas de desenvolvimento de RBC existentes, com a análise das principais características, assim como os pontos positivos e negativos do uso de cada ferramenta, e na Seção 2.3 é apresentada uma análise comparativa entre as ferramentas levantadas.

Na Seção 2.4 apresenta o *framework* jColibri 2, detalhando sua arquitetura, funcionamento e principais características.

Na Seção 2.5 é apresentada a estratégia de ensino que será utilizada na ferramenta possibilitando ao usuário, além de apenas desenvolver um sistema RBC, obter o aprendizado necessário sobre a técnica, utilizando efetivamente o ambiente.

2.1 RACIOCÍNIO BASEADO EM CASOS

Os sistemas de Raciocínio Baseado em Casos (RBC), ou *Case Based Reasoning* (CBR), surgiram de um modelo de armazenamento de informações na memória humana, sendo atualmente utilizados como uma técnica para resolução de novos problemas baseados em soluções resolvidas anteriormente.

A idéia básica desta técnica é a de buscar a solução para uma situação atual através da comparação com uma experiência passada semelhante. O processo característico de RBC consiste em: identificar a situação atual, buscar a experiência mais semelhante na memória e aplicar o conhecimento desta experiência passada na situação atual.

2.1.1 ORIGEM DOS SISTEMAS RBC

Os sistemas RBC tiveram sua inspiração inicial em 1977, quando Schank e Abelson no seu trabalho conhecido como Memória Dinâmica, propuseram que o conhecimento geral das pessoas sobre as situações está armazenado em *scripts* ou roteiros. Estes roteiros permitem que se criem expectativas sobre resultados esperados de ações e se faça inferências sobre relacionamentos casuais entre ações. Como exemplo Schank descreve o roteiro do restaurante, que demonstra um modelo computacional, que descreve o conjunto de ações realizadas em uma ida a um restaurante. (WANGENHEIM & WANGENHEIM, 2003)

Segundo Aamodt e Plaza (1994) outra linha de pesquisa que contribuiu para o desenvolvimento de sistemas RBC foi o Raciocínio por Analogia, como apresentado no trabalho de Gick e Holyoak, no início dos anos 80. No raciocínio analógico ocorre uma mudança de domínio durante a transformação da solução para um novo problema, porém para os sistemas RBC o domínio do problema deve ser exclusivamente sobre o mesmo domínio de aplicação.Também contribuíram para o desenvolvimento dos sistemas RBC pesquisas na área das Teorias da Formação de Conceitos, da Resolução de Problemas e da Aprendizagem Experimental no âmbito da filosofia e da psicologia.

Schank continuou a explorar que importância a memória de situações prévias, também conhecidas como MOP's (*Memory Organization Packets* - Pacotes de Organização de Memória) tem, tanto na resolução de problemas, como durante o aprendizado de situações. Esta teoria define que a compreensão, a recuperação e o aprendizado de situações são processos diretamente ligados (AAMODT & PLAZA, 1994).

O primeiro sistema realmente desenvolvido para funcionar como um sistema baseado em casos foi denominado de CYRUS, que é um sistema desenvolvido por Janet Kolodner, baseado no modelo de memória dinâmica de Schank e na teoria dos MOP's para aprendizagem e solução de problemas. CYRUS é um sistema de perguntas e respostas que integra o conhecimento obtido da descrição de várias viagens e reuniões do ex-secretário de estado dos Estados Unidos, Cyrus Vance. O objetivo do sistema era encontrar uma situação diplomática anterior semelhante a uma questão colocada ao sistema e propor a solução adotada por Cyrus Vance naquela ocasião (WANGENHEIM & WANGENHEIM, 2003).

Após CYRUS, segundo Aamodt e Plaza (1994), outros sistemas de RBC foram criados, como:

- PROTOS: Evidencia o conhecimento de casos específicos em uma estrutura representativa de casos não definidos.
- MEDIATOR: Opera sobre o domínio da solução de disputas entre diversos partidos, sugerindo compromissos. Caso uma proposta falhe em satisfazer todas as partes interessadas, gera novas propostas e grava a falha, evitando uma falha similar no futuro.
- HYPO: Atua na área de legislação de marcas e patentes, centralizado em casos de violação de patentes, como a divulgação de um segredo industrial, utilizando sua base de casos de precedentes para gerar argumentos plausíveis para o processo de defesa.
- CASEY: Aperfeiçoa o desempenho de sistemas baseados em conhecimento.
- CHEF: esenvolve novos pratos a partir de outros. São solicitados os objetivos e então se procura satisfazer estes, recuperando do banco a receita que atenda a maior parte dos objetivos e cria-se uma nova receita para a situação apresentada.

Segundo Wangenheim e Wangenheim (2003) os sistemas RBC estão se disseminando por todo o mundo, sendo que o número de trabalhos de pesquisa e de aplicações comerciais na área está

aumentando rapidamente, pois o número de sistemas de RBC comerciais, principalmente, cresceu muito nos últimos anos.

2.1.2 DEFINIÇÃO

RBC é uma técnica de Inteligência Artificial com 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 (AAMODT & PLAZA, 1994).

Para o desenvolvimento de sistemas RBC é necessário inicialmente utilizar soluções já aplicadas anteriormente com sucesso e determinar qual das experiências mais se assemelha ao problema atual. Para isso se torna necessário que estas experiências passadas sejam analisadas e armazenadas de maneira organizada. O sistema RBC é responsável por realizar a pesquisa nestas experiências armazenadas e verificar se existem casos semelhantes com as características do problema atual, podendo encontrar um ou mais casos, que serão utilizados para chegar a solução do problema. O sistema é capaz de localizar e encontrar partes de casos que não se adequem ao problema, criando um novo caso para uso posterior (LEE, 1998).

Desta forma, um sistema RBC pode ser dividido em quatro elementos básicos:

- Representação do Conhecimento: em um sistema de RBC, o conhecimento é representado principalmente em forma de casos, que descrevem experiências concretas.
- Medida de Similaridade: define como será calculada a similaridade entre a situação atual e um determinado caso na base de casos, sendo aplicada repetidamente, par a par, para todos os casos, chegando a um valor de similaridade. Estes valores e casos são ordenados e os mais similares são sugeridos como solução potencial para o problema presente.
- Adaptação: situações passadas representadas como casos dificilmente serão idênticas às do problema atual. Sistemas de RBC avançados têm mecanismos e conhecimento para adaptar os casos recuperados completamente, para verificar se satisfazem às características da situação presente.

- Aprendizado: para que um sistema se mantenha atualizado e evolua continuamente, sempre que ele resolver um problema com sucesso, deverá ser capaz de lembrar dessa situação no futuro como mais um novo caso.

Um caso é a forma de conhecimento contextualizado representando uma experiência que ensina uma lição útil. As lições úteis podem ser definidas como aquelas que têm o potencial para ajudar o raciocinador a alcançar uma meta ou um conjunto de metas ou advertem sobre a possibilidade de uma falha ou apontam para um problema futuro (KOLODNER, 1993).

Um caso é representado tipicamente pela descrição de uma situação juntamente com o resultado obtido durante a sua resolução, obtendo desta forma a associação dos dois conjuntos de informações: a descrição do problema e respectiva solução. Para que estes casos estejam a disposição para serem reutilizados, eles são organizados e armazenados em uma base de casos, formando um conjunto que geralmente contém experiências positivas descrevendo estratégias de solução que contribuíram com sucesso na resolução do problema descrito, de forma que possam ser reutilizadas. Experiências negativas, expressando tentativas frustradas de solução também podem ser armazenadas, com o objetivo de indicar problemas potenciais e prevenir a repetição de erros passados (WANGENHEIM & WANGENHEIM, 2003).

Os sistemas RBC possuem muitas características relevantes para sua utilização, porém um problema importante que deve ser tratado é o armazenamento e os cálculos necessários para recuperação dos casos na base de casos. Segundo Luger (2004), à medida que o sistema RBC vai adquirindo mais casos, ele se torna mais inteligente, capaz de resolver uma variedade de problemas-alvo, aumentando seu desempenho em resolução de problemas. O problema é quando a etapa de aprendizado é implantada e a base de casos aumenta, aumentando em conjunto o tempo necessário para recuperar e processar um caso.

Luger (2004) afirma que uma série de outros fatores pode causar o declínio da eficiência de grandes bases de casos, como a superposição de casos, ruído e a distribuição de tipos de problemas. Uma solução para este problema é armazenar apenas os “melhores” casos ou “protótipos”, excluindo aqueles que são redundantes ou que não são usados com freqüência.

Estes fatores mostram que a representação do conhecimento é um aspecto importante do RBC, e será tratado com maiores detalhes na Subseção 2.4.3. Na Subseção 2.4.4 são apresentadas

as métricas de similaridade para sistemas RBC e na Subseção 2.4.5 são apresentadas as etapas de um sistema RBC.

2.1.3 REPRESENTAÇÃO DOS CASOS

Aamodt e Plaza (1999, *apud* FERNANDES, 2003) afirmam que a representação de casos é uma tarefa complexa e importante para o sucesso ou fracasso do sistema RBC. O grande problema está no momento de decidir o que será armazenado e encontrar a estrutura ideal para descrever o conteúdo do caso.

Os dados que serão armazenados em cada caso dependem do domínio de aplicação e o objetivo do problema que se pretende resolver, tendo em comum para todas as aplicações a representação de uma situação experimentada.

Um caso é composto tipicamente de dois componentes principais (WANGENHEIM & WANGENHEIM, 2003):

- Descrição do problema: descreve o estado do mundo quando o caso ocorreu. Pode representar um problema que necessita ser resolvido ou uma situação que necessita ser interpretada, classificada ou compreendida. Esta descrição deve possuir informações suficientes para que seja possível julgar a aplicabilidade de um caso à nova situação.
- Solução: postula a solução derivada para aquele problema. A solução pode ser representada através de várias informações tais como, uma ação, um plano ao uma informação útil ao usuário.

Para Kolodner (1993) um caso ainda pode ser composto por um terceiro componente: resultado da aplicação. Este componente armazena o resultado obtido pela aplicação, informando se com o resultado foi ou não obtido sucesso, apontando para uma provável próxima solução.

Para representação do conhecimento existem diferentes maneiras de definir os formalismos com os quais será formulado o conhecimento para a solução de problemas do sistema, se tornando necessária a utilização de uma linguagem definida. Para o contexto do RBC, Wangenheim e Wangenheim (2003) citam diferentes formas que podem ser utilizadas para esta representação:

- Atributo-Valor: um item de dado pode ser representado por um par atributo-valor, onde o conjunto destes pares atributo-valor forma um caso. Este conjunto pode ser fixo para todos os casos ou variar entre casos individuais.

- Orientada a objetos: o conhecimento é organizado de maneira similar às linguagens de programação orientada a objetos, onde é descrito o domínio, particionando-o de acordo com seus objetos. Este tipo de representação é indicado para domínios de aplicações complexos, onde podem ocorrer casos com estruturas variáveis.

- Árvore e Grafos: os casos são representados através de grafos dirigidos, não dirigidos ou atributo relacional.

- Redes Semânticas: são um tipo específico de grafo similar a uma rede, onde a situação ou entidade a representar normalmente possui uma estrutura composta, com nodos representando unidades conceituais e arestas dirigidas representando relacionamentos entre estas unidades.

- Árvores K-D: o princípio desta representação é estruturar o espaço de busca com base na sua densidade observada e a utilização desta estrutura pré-compilada para a recuperação eficiente de casos de acordo com uma medida de similaridade dada.

Para Wangenheim e Wangenheim (2003) a utilização da representação atributo-valor traz vantagens sobre as demais representações por ser simples, de fácil implementação, simplificando a implementação de medidas de similaridade eficientes, sendo recomendada para representação de casos que tenham de lidar com grandes bases de casos.

Ao definir a forma de representação dos casos, um ponto que deve ser levado em consideração é a forma como serão organizados e recuperados estes casos na base de casos. Segundo Lee (1998) a organização dos casos na memória é escolhida conforme a natureza dos dados, a forma de representação dos casos e etapas de desenvolvimento do sistema RBC. Um estilo de organização normalmente utilizada em representação atributo-valor é através de banco de dados relacional, podendo também utilizar outras formas, tais como: memória plana, memória hierárquica, redes e árvores.

Para a recuperação dos casos torna-se necessária a utilização de índices. Os índices de um caso são combinações de seus atributos mais importantes, que permitem distinguí-lo de outros e

identificar casos úteis para uma dada descrição do problema (WANGENHEIM & WANGENHEIM, 2003). A indexação determina quais os atributos devem ser comparados para se avaliar a similaridade entre o caso de entrada e os casos da base (FERNANDES, 2003). Escolher os índices pode requerer uma interpretação do processo de elaboração durante o qual as características funcionais podem ser feitas de forma separada, devendo procurar formas de descrever ou representar o caso. Para qualquer caso similar precisa-se designar quais partes da descrição ou quais características atuarão como índices (KOLODNER, 1993).

2.1.4 INDEXAÇÃO

A indexação deve ser considerada conforme a estrutura e o conteúdo da memória. A memória é indexada para proporcionar uma recuperação e reutilização eficiente, ou seja, se define como índices os atributos que serão utilizados para realizar a comparação entre o caso e a situação presente, sendo estes atributos conhecidos como discriminantes.

Segundo Wangenheim e Wangenheim (2003) os índices de um caso são combinações de seus atributos mais importantes, que permitem distinguí-lo de outros e identificar casos úteis para uma dada descrição de problema.

Para Kolodner (1993), indexação de casos é a associação de rótulos em casos, de maneira a caracterizá-los para posteriormente recuperá-los em uma base de casos. Esta não é uma tarefa simples. Para construir uma boa coleção de índices para um conjunto de casos é necessário ser ter em mente a importância de um bom índice e como definí-lo.

Portanto, os índices são utilizados para determinar o grau de similaridade entre um caso e outro. A escolha dos índices deve ser realizada cuidadosamente. Características superficiais são facilmente extraídas de um caso, entretanto, estas características podem ser menos úteis do que índices mais complexos obtidos pela combinação e composição das características que distinguem os casos de cada lição que ele pode ensinar.

Características preditivas são combinações de descritores de um caso responsáveis pela solução, que influenciam no resultado ou caracterizam o problema. Índices devem ser mais abstratos do que os detalhes de um caso particular. Embora casos sejam específicos, índices para casos precisam ser escolhidos para que o caso possa ser usado amplamente em uma coleção de situações apropriadamente. Enquanto índices precisam ser geralmente aplicáveis, eles também

precisam ser suficientemente concretos, para que possam ser reconhecidos com pouca inferência. Este índices devem ser escolhidos para fazer os tipos de predição que seriam úteis em futuros raciocínios. Índices úteis são aqueles que rotulam um caso como sendo capazes de dar guias sobre as decisões que o motor de inferência irá tratar (SANTOS, 2004).

A seleção dos índices pode ser realizada manualmente ou automaticamente. A seleção manual analisa caso a caso para determinar quais características descritas que determinam as variações sobre as conclusões. A seleção automática busca quantificar as diferenças entre os casos e os relacionamentos entre o problema e soluções adotadas. Abel (1996) cita algumas formas de selecionar índices:

- Técnicas baseadas em explicação: os casos são analisados individualmente para determinar os elementos do problema que são utilizados para construir a solução. Esses elementos são utilizados como índices.
- Índices baseados em conhecimento do domínio: utilizando protocolos retrospectivos sobre os casos, são extraídas as correlações entre elementos e conclusões nos casos particulares e no domínio como um todo (processos abstratos). Esses elementos e processos são utilizados como índices.
- Análise matemática: todos os elementos do domínio e suas dimensões são analisados numericamente para identificar quais as feições que determinam ou influenciam as conclusões. Os elementos e valores computados são utilizados para construir os índices. São os métodos utilizados nos sistemas MEDIATOR e CHEF.
- Índices baseados nas diferenças entre os casos: o sistema analisa casos similares e os indexa especificamente nas características que os diferenciam, como no sistema CYRUS.
- Métodos de generalização: o método utiliza a definição de casos abstratos a partir dos elementos compartilhados entre diversos casos armazenados. Esses elementos são utilizados para a indexação dos casos abstratos, enquanto que as funções que os diferenciam indexam os casos individuais.
- Métodos de aprendizado indutivo: identificam os elementos que determinam as conclusões para serem utilizados como índices. Esses métodos são muito difundidos especialmente em variações do algoritmo para indução de regras ID3.

Existem métodos automatizados que auxiliam na escolha de bons índices, porém na prática, os sistemas cujos índices foram definidos manualmente tendem a ter melhor desempenho do que aqueles puramente processados.

2.1.5 ETAPAS DE UM RBC

O modelo de funcionamento de um sistema RBC mais aceito é o Ciclo de RBC, que engloba um ciclo de raciocínio contínuo composto por quatro tarefas (AAMODT & PLAZA, 1994):

- Recuperação: tem como objetivo encontrar um caso ou um pequeno conjunto de casos na base de casos que contenha uma solução útil para o problema ou situação atual. Para esta recuperação torna-se necessário comparar a descrição do problema atual com os problemas armazenados na base de casos, aplicando uma medida de similaridade. Esta comparação é realizada a partir dos índices dos casos, que informam as principais características de cada caso, permitindo a recuperação eficiente de casos que tenham a maioria das características em comum com o problema atual.
- Reutilização: após a recuperação de um caso adequado, a solução sugerida por este caso é objeto de uma tentativa de reutilização de conhecimento para a solução do problema atual. O objetivo principal da reutilização consiste na adaptação da solução do caso anterior ao caso atual, trazendo flexibilidade aos sistemas de RBC. Em determinados domínios de aplicação de RBC esta adaptação não é aplicada, sendo suficiente que seja realizada a cópia da solução do caso encontrado para o caso atual e se aplique esta solução ou então que se adapte o caso manualmente.
- Revisão: quando a solução gerada para um caso na fase de reutilização não é correta, surge a oportunidade para o aprendizado a partir desta falha, sendo então necessária a revisão do caso. Esta etapa consiste em avaliar criteriosamente a solução gerada pela reutilização, onde caso esta avaliação seja considerada correta, segue-se para etapa de retenção, caso contrário, repara-se a solução para o caso, utilizando-se de conhecimento específico sobre o domínio da aplicação ou informações fornecidas pelo usuário.
- Retenção: o objetivo de se reter continuamente o conhecimento toda vez que um novo problema é resolvido é o de constantemente atualizar e estender a base de casos, permitindo que o sistema RBC incremente continuamente seu conhecimento tornando-se um solucionador de problemas mais poderoso ao longo de sua utilização. Retenção

equivale a aprendizado de novos casos ou de conhecimentos relevantes abstraídos de casos individuais. O aprendizado efetivo em RBC requer um conjunto de métodos bem elaborado, de forma a extrair conhecimento relevante da experiência passada, indexar este conhecimento para uso posterior e integrar casos em uma estrutura de conhecimento existente.

A Figura 1 ilustra o funcionamento deste ciclo.

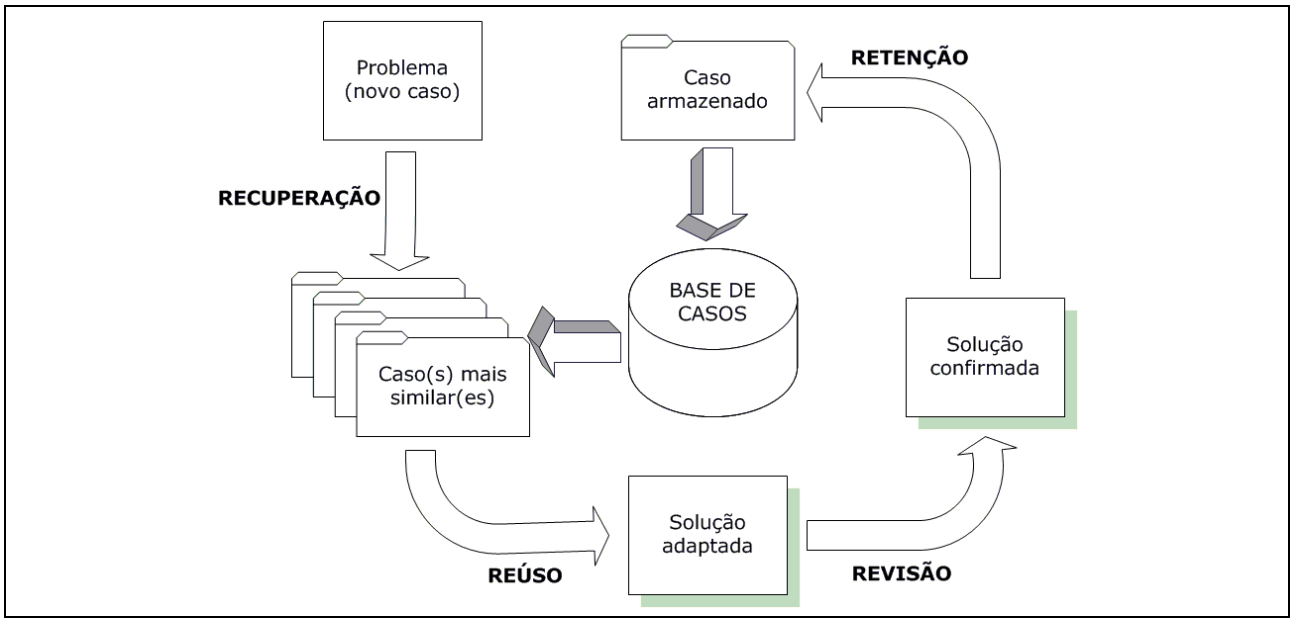


Figura 1. Ciclo do Raciocínio Baseado em Casos
Fonte: Adaptado de Wangenheim e Wangenheim (2003)

Todas as etapas relacionadas no ciclo não devem ser construídas isoladamente por estarem diretamente ligadas. Quando iniciado, o encadeamento harmônico do ciclo representa o sucesso ou o fracasso de um sistema RBC.

2.1.6 RECUPERAÇÃO DOS CASOS

A partir de um problema a ser resolvido (problema de entrada), a etapa de recuperação consiste em fazer uma busca na memória de casos e selecionar quais poderão ser aproveitados. A busca por casos é feita por algoritmos que selecionam casos com determinada similaridade com relação ao problema de entrada.

A etapa de recuperação envolve algumas tarefas, a saber: similaridade, métrica da similaridade, recuperação de casos e seleção do *best match*. Segundo Aamodt e Plaza (1994), há três maneiras de recuperar casos:

- buscando diretamente os índices das características;
- fazendo uma busca numa estrutura de índices; e
- fazendo a busca em um modelo de conhecimento mais amplo.

Para recuperação dos casos, os sistemas de RBC procuram em locais de memória diretamente através de heurísticas *match* e *ranking*, acessando casos que podem ser úteis. Segundo Kolodner (1993), *match* é um processo que compara dois casos entre si e determina o grau de similaridade entre eles e *ranking* ordena os casos *partially-matching* conforme sua utilidade, determinando quais são os melhores entre os outros. Sendo assim, este processo funciona em duas etapas:

- recuperação de casos da base: o objetivo deste processo é recuperar um conjunto de casos que apresentem potencialidade de serem aproveitados nas próximas etapas a serem executadas. Um caso será recuperado em função de aspectos que caracterizam uma determinada situação. Estes aspectos em geral são os elementos que compõem as chaves de indexação dos casos armazenados;
- seleção dos melhores casos: entre os casos selecionados anteriormente, escolhe-se os melhores. Este processo seleciona um caso - ou mesmo um pequeno conjunto de casos - como mais promissor dentre os recuperados no processo anterior.

A habilidade de distinguir quais dos vários casos parcialmente similares (*partially-matching*) têm o potencial de ser mais útil que outros é a chave para fazer sistemas RBC funcionarem.

Segundo Kolodner (1993), geralmente a eficiência da recuperação depende do valor e da complexidade da comparação da similaridade que deve ser realizada, e a precisão depende dos índices. A meta das estratégias de buscas é selecionar o maior conjunto potencial de casos relevantes da base, e ao mesmo tempo certificar que pelo menos alguns destes casos têm o potencial de serem mais úteis que outros do conjunto. As estruturas organizacionais particulares e os algoritmos são apropriados para qualquer aplicação dependendo amplamente do número de casos na

base de casos, da complexidade dos índices e do número de tarefas diferentes que a base de casos suporta, e a variabilidade dos índices em todas estas tarefas.

As principais formas de recuperação de casos são (JULIO, 2005; KOLODNER, 1993; KOSLOSKY, 1999; FERNANDES, 2003):

- **Flat memory, serial search:** os casos são armazenados sequencialmente em listas simples, *array* ou arquivo. Os casos são recuperados aplicando uma função de comparação seqüencialmente para cada caso do conjunto, mantendo o grau de correspondência do caso atual com o pesquisado, retornando os casos com maior similaridade. Nesta estrutura, todos os casos da base são analisados e o algoritmo de recuperação é relativamente simples, porém se a base de casos possuir muitos casos o desempenho será degradado.

- **Shared feature networks (Redes de Características Compartilhadas):** os casos são organizados de maneira hierárquica para que se considere somente pequenos conjuntos de informações durante a recuperação, tendo como maior vantagem a recuperação que é realizada de maneira mais eficiente quando comparado com listas, porém a inclusão de novos casos na base é complexa, pois necessita alocar o novo caso de maneira correta.

- **Discrimination networks (Redes discriminantes):** são utilizadas árvores (redes) com características particionadas que agrupam casos similares. É primordial discriminar os casos, e o agrupamento surge como efeito desta discriminação. As vantagens deste método são sua maior eficiência na recuperação em comparação as Redes de Características Compartilhadas e a compreensão da conexão entre índices, sendo que a organização da árvore é indutiva. Como desvantagem, tem-se as mesmas das Redes de Características Compartilhadas onde não fica claro como são tratadas as informações perdidas.

- **Redundant discrimination networks:** organizam os itens usando diferentes Redes Discriminantes, cada uma com uma ordem diferente das questões, sendo a pesquisa realizada em paralelo. Se a resposta a uma questão em uma árvore não é conhecida, a pesquisa nesta árvore é interrompida, mas continua em outra árvore. A principal vantagem é que as generalizações são formadas ao adicionar novos casos à árvore.

Pode-se citar como desvantagem as mesmas encontradas nas Redes Discriminantes, que necessitam de espaços adicionais para o armazenamento das informações na árvore e os procedimentos para inserção de novos casos são complexos.

- **Flat memory, parallel search:** utiliza processamento paralelo para recuperação das informações da base de casos, aplicando funções de casamento em todos os casos, na ordem dos casos, tendo como vantagem o desempenho do processamento paralelo, onde toda a base de casos é pesquisada durante o processo de recuperação.Sua desvantagem é a necessidade de um hardware para processamento paralelo para implementar a solução.

- **Hierarchical memory, parallel search:** organiza a memória de forma hierárquica, sem índices, de maneira que faça sentido, sendo que as buscas são realizadas em paralelo através de múltiplos processamentos. As vantagens deste método são a recuperação eficiente e as generalizações acessíveis pelos mesmos algoritmos que acessam os casos.

- **Inductive Retrieval (Recuperação indutiva):** é uma técnica desenvolvida por máquinas de aprendizagem buscando extrair regras ou árvores de decisão de um caso. Em sistemas RBC a base de casos é analisada por um algoritmo de indução para produzir uma árvore de decisão que classifica os casos, sendo o algoritmo mais utilizado chamado ID3.

Os métodos de recuperação estão diretamente relacionados com o desempenho dos sistemas de RBC, pois definem o tempo necessário para recuperar e armazenar as informações dos casos.

2.1.7 MEDIDAS DE SIMILARIDADE GLOBAL

Similaridade é o ponto principal de um sistema RBC, pois a similaridade é o raciocínio que dá suporte ao sistema. A avaliação da similaridade do caso a ser solucionado se faz comparando-se com os casos candidatos, pois o que torna um caso similar a outro é a semelhança das características que irão representar realmente o conteúdo e o contexto da experiência (DELPIZZO, 1997 *apud* FERNANDES, 2003).

Segundo Lee (1998), um caso será similar ao outro quando as características que representam realmente o conteúdo e o contexto do mesmo forem semelhantes. As características relevantes do problema são aquelas que, quando combinadas entre si, determinam a sua solução.

Burkhard (1998, *apud* WANGENHEIM & WANGENHEIM, 2003) afirma que, para a determinação da similaridade de um sistema de RBC, as seguintes premissas têm de ser satisfeitas:

- a similaridade entre a questão atual e o caso implica utilidade;
- a similaridade é baseada em fatos *a priori;* e
- como casos podem ser mais ou menos úteis em relação a questão, a similaridade precisa prover uma medida.

A maneira mais conhecida de formalizar o conceito de similaridade é a definição de uma medida numérica de distância ou similaridade. Lee (1998) afirma que:

> A métrica da similaridade tem por principal objetivo dar um valor numérico para similaridade entre dois casos, sendo todos os casos da memória avaliados comparativamente ao caso de entrada. Supondo-se uma função binária que retorne 0 ou 1 para funções diferentes ou iguais entre casos, a soma destes resultados já será uma medida de similaridade. Na prática, porém, as aplicações vão além, estabelecendo-se um grau de similaridade entre os atributos de determinadas dimensões, um número do intervalo [0,1].

Diferentes maneiras podem ser utilizadas para se obter a similaridade entre casos, tendo cada uma sua fórmula definida. Segundo Julio (2005) e Wangenheim e Wangenheim (2003), entre as principais métricas de similaridade estão: a) Vizinho mais próximo; b) Distância Euclidiana; c) Distância de Manhattan; d) Distância de Chebychev; e) Coeficiente de Casamento Simples; f) Modelo de contraste; e g) Modelo do vetor. Cada uma destas métricas é apresentada nas seções seguintes.

2.1.7.1 VIZINHO MAIS PRÓXIMO

A métrica do vizinho mais próximo utiliza um sistema geométrico de coordenadas, onde para cada atributo pode-se obter a medida de distância entre o novo problema e os casos existentes, podendo ainda cada atributo possuir um peso diferenciado. Esta medida deve ser multiplicada pelo respectivo peso, e então calculada a somatória de todos os atributos. Este método é considerado bastante simples, não necessitando de muitos cálculos para compreensão.

Vizinho mais próximo é uma técnica simples, que pode ser usada para buscar casos similares na base de casos, a partir de um novo caso, não requer muitos para sua compreensão (JULIO, 2005). Segundo Fernandes (2003), a técnica do vizinho mais próximo é utilizada pela grande maioria das ferramentas de RBC. A fórmula da similaridade pelo vizinho mais próximo é apresentada na Equação 1.

$$\text{Similaridade (q,c)} = \sum_{i=1}^{n} f(q_i, c_i) * w_i \qquad \text{Equação 1}$$

Sendo:

q = Novo caso.

c = Casos existentes na base de casos.

n = Número de atributos.

i = Atributo individual variando de 1 a n.

f = Função de similaridade para o atributo i nos casos N e F.

w = Peso do atributo i.

A função de similaridade (f) deve ser definida em relação a cada tipo específico de atributo, dependendo altamente do domínio de aplicação. Esta definição se torna necessária, pois os atributos de um caso podem ser representados através de diferentes tipos, como por exemplo números, símbolos, conjuntos, intervalos, *strings*, etc.

As similaridades são geralmente normalizadas em uma faixa de 0 e 1, onde 0 é a dissimilaridade total e 1 a coincidência absoluta, ou através de porcentagens, onde 100% é o casamento exato. A Normalização é realizada com a divisão do valor de similaridade pela soma total dos pesos dos índices. Desta forma surge a fórmula de similaridade do vizinho mais próximo ponderado, conforme apresentado na Equação 2, seguindo a mesma definição de variáveis da Equação 1.

$$\text{Similaridade (q,c)} = \frac{\sum_{i=1}^{n} f(q_i, c_i) * w_i}{\sum_{i=1}^{n} w_i} \qquad \text{Equação 2}$$

A função de similaridade é realizada para cada atributo do caso, e a equação de similaridade é realizada para toda a base de casos, obtendo o *ranking* de cada caso. Este *ranking* estabelece uma medida de similaridade no intervalo entre 0 e 1, onde 0 representa o caso menos similar e 1 representa o caso totalmente similar.

2.1.7.2 DISTÂNCIA EUCLIDIANA

A distância euclidiana apresenta a distância real entre dois pontos em um espaço multidimensional. Na Equação 3 é apresentada a fórmula euclidiana para índices que não possuem pesos.

$$D(q,c) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \qquad \text{Equação 3}$$

Com a introdução de pesos para os índices, pode-se estender esta métrica para a distância euclidiana ponderada, conforme apresentada na Equação 4.

$$D(q,c) = \sqrt{\sum_{i=1}^{n} w_i * (q_i - c_i)^2} \qquad \text{Equação 4}$$

A distância euclidiana é a raiz quadrada da soma dos quadrados das diferenças de valores para cada variável. Ou seja, apresenta a distância reta (direta) entre o caso atual e o caso armazenado na base de casos, resultando em 0 quando os casos comparados são idênticos e tende a infinito quando os casos são totalmente diferentes.

2.1.7.3 DISTÂNCIA DE MANHATTAN

A Distância de Manhattan (Métrica do Quarteirão) é uma medida neutra, uma vez que pondera todas as diferenças de forma idêntica. Ela é considerada uma simplificação da distância Euclidiana, e, por isso, é uma medida mais simples e de fácil implementação. É indicada para aplicações em tempo real devido a sua simplicidade.

$$D(q,c) = \sum_{i=1}^{n} |q_i - c_i| \qquad \text{Equação 5}$$

Esta distância, apresentada através da Equação 5, não apresentada a distância reta entre dois pontos em um plano. Esta distância faz analogia aos quarteirões da ilha de Manhattan, sendo capaz de modelar as trajetórias, “por linhas quebradas”, dos cidadãos e dos veículos que se deslocam entre quarteirões, ao longo dos eixos de ruas e avenidas.

Segundo Miranda (1999), na distância de Manhattan, a menor distância entre dois pontos de um plano não é a linha reta. A distância não é medida como o vôo de um pássaro, mas como a viagem de um táxi numa cidade, cujas ruas estendem-se vertical e horizontalmente em uma quadra ou malha urbana, que convenientemente pode ser associada ao plano euclidiano.

2.1.7.4 DISTÂNCIA DE CHEBYCHEV

Esta distância é considerada apropriada, segundo Fernandes, Miranda e Silva (2003), para casos onde exista a necessidade de definir dois objetos como diferentes, caso sejam diferentes em qualquer dimensão. Na Equação 6 é apresentada a fórmula da distância de Chebychev.

$$d(q,c) = Máximo|\, q_i - c_i \,| \qquad \text{Equação 6}$$

Um ponto importante é que esta medida de distância é muito sensível a medidas periféricas (distâncias com maior valor).

2.1.7.5 COEFICIENTE DE CASAMENTO SIMPLES

O coeficiente de casamento simples utiliza como base a Distância de Hamming que é definido como o número de bits divergentes em dois vetores x e y de mesmo tamanho. Essa métrica utiliza como base uma tabela de contingência, que é adequada para representação de atributos binários e simétricos.

Na tabela de contingência cruzam-se os valores do caso atual e o da base de casos, quantificando os atributos iguais e diferentes, obtendo-se desta forma quatro quadrantes: A e D representando os valores que correspondem e B e C representando os valores divergentes, conforme apresentado na Tabela 1.

Tabela 1. Tabela de Contingências para valores binários

x / y	**1**	**0**
1	A	B
0	C	D

Fonte: Adaptado de Wangenheim e Wangenheim (2003).

Desta maneira se expressa formalmente os quadrantes através da Equação 7.

$$a = \sum_{i=1}^{n} (x_i\, y_i) \qquad b = \sum_{i=1}^{n} (x_i\, \overline{y_i}) \qquad c = \sum_{i=1}^{n} (\overline{x_i}\, y_i) \qquad d = \sum_{i=1}^{n} (\overline{x_i}\, \overline{y_i}) \qquad \text{Equação 7}$$

Assim, no coeficiente de casamento simples utiliza-se simplesmente o número de valores correspondentes entre o caso atual e o caso armazenado na base de casos como medida de similaridade, conforme apresentado na Equação 8.

$$sim(q, c) = 1 - \frac{b + c}{n} = \frac{a + d}{a + b + c + d} \qquad \text{Equação 8}$$

A Equação 8 apresenta os atributos correspondentes e divergentes com pesos iguais, porém ainda pode-se definir uma importância maior para os atributos correspondentes e divergentes, incluindo um peso α, onde 0 ≤ α ≤1, conforme apresentado na Equação 9.

$$sim(q,c) = \frac{a(a+d)}{a(a+d) + (1-a)(c+b)} \qquad \text{Equação 9}$$

Sendo o valor de α :

- Igual ½: Se obtem o peso igual para os atributos, conforme apresentando na Equação 7.
- Menor que ½: Destacam-se os casos com maior divergência entre os atributos;
- Maior que ½ : Destacam-se os casos com maior correspondência entre os atributos.

2.1.7.6 MODELO DE CONTRASTE

Segundo Wangenheim e Wangenheim (2003), a regra de contraste descreve a similaridade entre um caso C e uma situação atual. Este modelo expressa a similaridade de dois objetos através de um contraste linear de diferenças ponderadas entre seus atributos comuns e discordantes.

Três conjuntos de atributos diferentes são criados:

1. S ∩C: Atributos disponíveis na situação S e no caso C;
2. C – S: Atributos somente disponíveis no caso; e
3. S – C: Atributos somente disponíveis na situação.

Os conjuntos de atributos individuais são ponderados com *a*, *b*, *c* e cada atributo possível no caso ou situação é ponderado, por sua vez, com $w_i \in R$. A medida de similaridade deste método é apresentada na Equação 10.

$$sim(S,C) = \left(a \sum_{i \in (S \cap C)} w_i\right) - \left(b \sum_{i \in (C-S)} w_i\right) - \left(c \sum_{i \in (S-C)} w_i\right) \qquad \text{Equação 10}$$

Desta forma, a subdivisão em diferentes conjuntos de atributos permite a manipulação explicita e diferenciada de informações desconhecidas e de atributos contraditórios e correspondentes, ao associar-se cada um deles a um peso diferente. Dependendo da escolha, pode-se ponderar valores correspondentes de forma mais forte que os contraditórios, ou vice-versa.

Como em toda medida de similaridade, os pesos dependem do contexto e podem ser alterados durante o ciclo de vida de um sistema RBC. Assim, à medida que a base de casos cresce, pode-se tornar a estratégia mais pessimista, alterando o valor dos pesos, para evitar a apresentação de conjuntos-solução muito grandes.

2.1.7.7 MODELO DE VETOR

Este método é utilizado para recuperação de informações em bases de casos textuais onde existe dificuldade em calcular a similaridade entre palavras. Desta forma, este método busca avaliar os casos buscando os casos considerados parcialmente iguais ao caso atual, através de termos considerados índices, que são aqueles que não estão presentes em *Stop List*, que contém todos os termos que serão desconsiderados na comparação. Todos os outros termos são considerados relevantes para representação na comparação.

Por classificação em ordem decrescente de grau de similaridade são recuperados os casos, os quais o modelo considera parcialmente iguais aos termos da dúvida (BAEZA-YATES & RIBEIRO-NETO, 1999).

A utilização da técnica do Modelo de Vetor aplica-se a problemas com dificuldade em calcular a similaridade entre *strings*, pois esta demandaria uma comparação semântica entre as palavras, tornando o processo muito complexo. Assim sugere-se substituir as *strings* por valores simbólicos sempre que possível, facilitando o cálculo da similaridade através de outras técnicas, tais como, a do Modelo de Vetor.

O Modelo de Vetor é definido pela indicação de pesos não-binários para os termos considerados como índices nos atributos de cada caso. Estes pesos de termos são utilizados para computar o grau de similaridade entre cada documento armazenado e a dúvida do usuário.

Por classificação em ordem decrescente de grau de similaridade são recuperados os casos, os quais o modelo considera parcialmente iguais aos termos do caso atual. Os termos considerados índices são aqueles que não estão presentes na *Stop List* - lista que contém todos os termos que

serão desconsiderados na comparação. Todos os outros termos são considerados relevantes para representação do conhecimento do caso atual ou do caso da base de casos.

Segundo Baeza-Yates e Ribeiro-Neto (1999) o modelo de vetor usa, para avaliar o grau de Similaridade (Sim) do documento, representado por um vetor **d** = [d_1, d_2, ... , d_n], com a consulta, representada por um vetor **q** = [q_1, q_2, ..., q_n]. A correlação entre os vetores d e q é ilustrada na Figura 2. Esta correlação pode ser quantificada pelo cosseno do ângulo entre estes dois vetores.

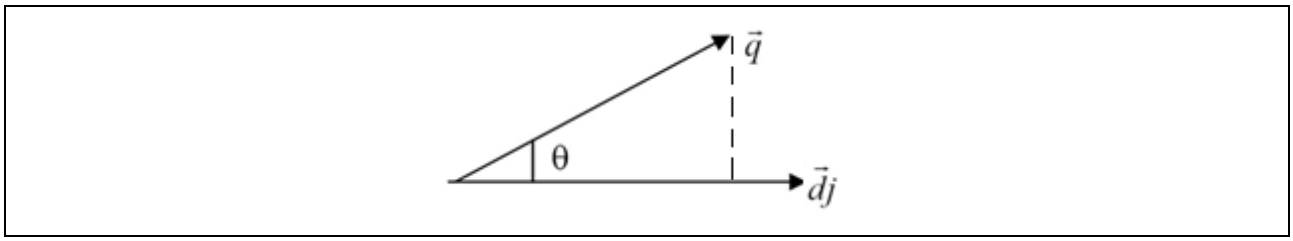


Figura 2. O cosseno de θ é adotado como Sim(dj,q)
Fonte: Adaptado de Lima e Rosatelli (2004)

Seguindo estas definições, chega-se a equação do modelo do vetor, conforme apresentado na Equação 11.

$$sim(\mathbf{d},\mathbf{q}) = \frac{\mathbf{d}\cdot\mathbf{q}}{|\mathbf{d}|\cdot|\mathbf{q}|} = \frac{\sum_{i=1}^{n}(d_i q_i)}{\sqrt{\sum_{i=1}^{n} d_i^2}\sqrt{\sum_{i=1}^{n} q_i^2}} \qquad \text{Equação 11}$$

Onde *n* corresponde ao número total de termos indexados no sistema.

Essa relação define o cosseno do ângulo entre os vetores dos termos, com valores entre 0 e 1. No modelo de vetor os termos são construídos utilizando-se uma medida para selecionar os termos. Essas medidas calculam o peso dos termos utilizados em um documento.

Para Baeza-Yates e Ribeiro-Neto (1999) no modelo de vetor, o peso dos termos é quantificado medindo-se a freqüência bruta da ocorrência de um termo dentro de um documento. A freqüência desse termo é usualmente denominada fator *tf* (freqüência do termo) e oferece uma medida de como esse termo descreve o conteúdo do documento. Além disso, a pesagem dos termos é quantificada através da medida inversa da freqüência de um termo entre uma coleção de

documentos. Este fator costuma ser denominada *como freqüência inversa de documentos ou* fator *idf.* A motivação para utilização do fator *idf* é que termos que aparecem em muitos documentos não são muito úteis para distinguir um documento relevante de um documento não relevante.

Neste método a utilização de um termo é definida através da freqüência que um termo é usado em um documento para calcular a informação local. A freqüência normalizada *f* de um termo em um documento é apresenta na Equação 12.

$$f = \frac{freq}{max\ freq} \qquad \text{Equação 12}$$

Onde *freq* é a freqüência de um determinado termo no documento e *Max freq* é o número máximo de vezes que um termo aparece no texto do documento, e é calculado sobre todos os termos que são mencionados. Também é utilizado como método de definição de peso um valor que varia inversamente como o número de documentos na coleção de documentos que utiliza o termo (*idf*)para calcular a informação global, e é representado pela Equação 13.

$$idf = \log\frac{N}{n} \qquad \text{Equação 13}$$

Onde, N é o número total de documentos da base de casos e *n* o número de documentos em que determinado termo aparece. O método mais conhecido de definição de peso dos termos é definido por Baeza-Yates e Ribeiro-Neto (1999) através da Equação 14.

$$\text{Peso do termo} = f * idf \qquad \text{Equação 14}$$

Teoricamente o modelo de vetor tem a desvantagem de que se assume que os termos índices são mutuamente independentes, ou seja, estes índices podem ser considerados individualmente sem levar em conta o contexto. Entretanto, na prática, considera-se que a dependência de termos pode ser uma desvantagem pois devido a localização de algumas dependências, sua aplicação indiscriminada em todos os casos da base pode ser um fato prejudicial à performance de recuperação (BAEZA-YATES E RIBEIRO-NETO, 1999).

Todas as métricas de similaridade apresentadas até o momento são utilizadas para comparação entre os casos da base e o caso atual. Porém, além desta comparação torna-se necessário a comparação entre cada atributo do caso, sendo necessário desta forma a utilização de uma métrica de similaridade local. Desta forma na sub-sessão seguinte são apresentadas as medidas de similaridade local.

2.1.8 MEDIDAS DE SIMILARIDADE LOCAL

Para o exame compreensivo da similaridade entre uma questão dada e os casos da base de casos, a similaridade local entre atributos específicos pode ser considerada ao se computar a similaridade global.

Para Wangenheim e Wangeheim (2003), como conseqüência, um caso com valores de atributo diferentes, mas que podem ainda ser similares ao procurado, não é passível de ser distinguido de outro, cujos valores são completamente diferentes. A consideração de similaridades locais permite a integração das similaridades entre atributos isolados ao cálculo da similaridade global, tornando-a muito mais sensitiva.

Estas medidas de similaridade devem ser definidas em relação ao tipo específico de um atributo. Outro ponto importante é que a similaridade local deve ser definida no contexto de aplicação específico, pois a similaridade entre dois valores pode variar entre diferentes sistemas.

As funções de similaridade local definem um valor entre o intervalo 0 – 1, que determina a similaridade de um atributo entre dois casos distintos. Cada função de similaridade pode ser utilizada para diferentes tipos de atributos dos casos, devendo ser definida a mais adequada para cada situação.

Entre as principais medidas de similaridade local utilizadas para tipos de atributos numéricos (inteiros e/ou reais) estão (WANGENHEIM & WANGENHEIM, 2003; KOLODNER, 1993; FERNANDES, 2003):

- Diferença entre valores: atributos podem ser descritos por números e o módulo da diferença destes valores pode ser considerado uma medida de similaridade.

- Função degrau (*Threshold*): é utilizada quando um atributo de um caso é totalmente útil ou totalmente inútil somente com relação ao valor da consulta. A função degrau

calcula uma similaridade de 1 se a distância entre dois valores for menor que o limiar definido para a função, senão a função retorna similaridade 0. Esta função também se aplica a valores booleanos, retornando similaridade 1 para valores iguais e 0 para valores diferentes.

- Função linear: é adequada para a maioria dos tipos numéricos, funcionando com a idéia de que a similaridade cresce com o decréscimo da distância entre os dois valores, ponderada pelo tamanho do intervalo assumido pelo domínio dos valores do atributo.

- Função assintótica ou escalar: a similaridade entre dois atributos pode utilizar a semântica de que diferenças entre dois valores causam uma redução assintótica da similaridade, sendo aplicadas neste caso funções polinomiais ou de outra natureza.

- Intervalo: a similaridade entre os atributos é considerada verificando se a diferença dos valores encontra-se dentro do intervalo, levando em conta os limites determinados.

Entre as principais medidas de similaridade local utilizadas para tipos de atributos textuais (caracteres e palavras) estão (WANGENHEIM & WANGENHEIM, 2003; KOLODNER, 1993; FERNANDES, 2003):

- Símbolo ordenado: representam valores simbólicos em uma determinada ordem, onde cada símbolo possui uma faixa de valores numéricos. A similaridade é verificada através dos valores e extraída a similaridade a partir do símbolo que o valor corresponde.

- Matriz de similaridade: é utilizada para símbolos não ordenados, onde cria-se uma matriz com os valores dos símbolos representados nas linhas e colunas, e nas células desta matriz encontram-se os valores de similaridade. Para verificação deste valor de similaridade é realizado o cruzamento do valor do atributo do caso atual com os casos da base de casos. No caso de dois atributos iguais aplica-se o valor de similaridade 1 e para os demais casos aplica-se um valor entre 0 e 1.

- Conjunto: neste caso os valores do atributo são interpretados como conjuntos, com seus valores mínimos e máximos definidos pela cardinalidade atribuída ao atributo.

Os valores de similaridade são calculados com base na identificação dos melhores casamentos para cada elemento do conjunto na consulta com o caso examinado.

- Símbolos taxonômicos: uma taxonomia é uma árvore n-ária na qual os nodos representam valores simbólicos descrevendo o relacionamento entre os valores e sua posição de taxonomia. Existem algumas formas de calcular a similaridade, entre elas pode-se utilizar para o cálculo o tamanho do caminho entre os objetos em questão na taxonomia.

- Distância de Hamming: também é utilizada para textos ou palavras, sendo definida como sendo o número de bits divergentes em dois vetores *x* e *y* de mesmo tamanho.

Entre as principais medidas de similaridade local utilizadas para textos (conjunto de palavras) estão (WANGENHEIM & WANGENHEIM, 2003; KOLODNER, 1993; FERNANDES, 2003):

- Correspondência exata: textos são similares se escritos da mesma forma.

- Correção ortográfica: comparando o número de caracteres que são idênticos, ponderado pelo número total de caracteres do texto de entrada.

- Contagem de palavras: contando o número de palavras idênticas entre dois casos.

- Taxa da maior palavra comum: onde a taxa da soma do tamanho das palavras comuns em relação ao tamanho total da consulta é computada.

Um ponto importante que ainda pode ser levantado quanto ao cálculo da similaridade entre casos é o estabelecimento de um limiar de recuperação dos casos, onde esta medida pode ser estabelecida informando o número absoluto de casos a serem recuperados ou também através de um limite de similaridade que se torne necessário ou eficiente para o experimento que se deseja realizar.

2.1.9 REUTILIZAÇÃO E TÉCNICAS DE ADAPTAÇÃO

Após selecionar os casos da base que são mais similares ao caso atual, a solução deste caso é objeto de uma tentativa de reutilização. Durante este processo pode-se aplicar uma adaptação do caso anterior ao caso atual, modificando determinados atributos que podem melhorar a solução apresentada.

Segundo Kolodner (1993), o fato de nenhum problema do passado ser exatamente igual a um problema atual, faz com que soluções passadas usualmente sejam adaptadas para solucionar novos problemas. A adaptação pode ser uma simples substituição de um atributo da solução por outro ou então, uma complexa e total modificação na estrutura da solução.

Para Julio (2005), a adaptação tem um papel fundamental na flexibilidade dos sistema de RBC, e sua capacidade de resolver novos problemas depende de sua habilidade em adaptar casos recuperados a novas circunstâncias e em sua habilidade de consertar soluções que falham ao serem aplicadas.

De acordo com Lagemann (1998), o processo de adaptação pode ser realizado de diversas formas:

- Inclusão de um novo comportamento à solução recuperada;
- Eliminação de um comportamento da solução recuperada; e
- Substituição de parte de um comportamento.

Para esta adaptação orientada à solução existe uma série de estratégias básicas, conforme apresentam Wangenheim e Wangenheim (2003), variando desde técnicas simples de estratégias básicas à adaptações bastante complexas, sendo que as principais exploradas são:

- Adaptação nula: é a mais simples das adaptações, onde nada no caso é adaptado, fazendo com que o caso recuperado da base seja apresentado diretamente ao usuário, deixando por conta do usuário as adaptações necessárias.
- Adaptação transformacional: significa que a solução do caso similar recuperado é transformada em uma nova solução, satisfazendo o novo problema com a aplicação de conhecimento específico ao domínio do problema, podendo ser aplicado ao novo

caso uma estrutura de regras ou se basear em casos da própria base de casos para sugerir uma alternativa.

- Adaptação gerativa: considerada como o método de adaptação mais complexo, onde para sua aplicação necessita-se armazenar junto ao caso o processo que levou a esta solução, incluindo informações sobre decisões tomadas e suas justificativas, aplicando este processo de solução do problema ao contexto do problema atual, sendo estas mesmas decisões testadas na solução do problema atual.

- Adaptação composicional: neste método são realizadas composições de solução adaptadas de vários casos anteriores, combinando para produzir uma solução composta, sendo possível quando a solução é composta de diferentes partes que possam ser adaptadas de forma mais ou menos independente.

- Adaptação hierárquica: os casos são armazenados em vários níveis de abstração, e a adaptação é realizada de cima para baixo, podendo reutilizar um único caso ou reutilizar casos diferentes de diferentes níveis de abstração, para refinar diferentes aspectos da solução.

Segundo Wangenheim e Wangenheim (2003), a maioria dos sistemas de RBC comerciais trabalha apenas com adaptação nula, pois RBC é considerada uma técnica poderosa para solução de problemas, mesmo não integrando técnicas de adaptação.

2.1.10 REVISÃO

Após a recuperação do caso na base de casos e sua adaptação, é necessário que a solução proposta seja avaliada. Caso a solução proposta não venha produzir um resultado satisfatório, então esta solução deve ser reparada para que uma nova solução seja gerada. Após encontrar a solução correta para o caso, a experiência obtida deve ser aprendida, sendo armazenada para uso futuro. Estes processos podem ser vistos como obtenção da experiência e os processos de recuperação e adaptação podem ser vistos como a aplicação da experiência adquirida.

De acordo com Lewis (1995 *apud* SANTOS, 2004) existem, de modo geral, três modelos de execução para uma solução: execução manual, execução sem supervisão e execução supervisionada.

Na execução manual, o usuário do sistema é responsável por interpretar a solução proposta e decidir se ela deve ou não ser executada. Ocorre na maior parte dos sistemas de RBC, em que o sistema somente sugere, com base na experiência, no processo de recuperação e no processo de adaptação, uma boa solução para o problema, que é executada pelo usuário.

Em alguns domínios, porém, quando uma solução é expressa em um programa de computador, um sistema de RBC pode ter a capacidade de executar a solução que ele propõe, realizando-a automaticamente sem intervenção ou controle humano. Quando isso ocorre, é formado um ciclo fechado de solução de problemas sem intervenção humana, em que o problema é submetido ao sistema, um caso similar é recuperado e sua solução é adaptada para a situação corrente, a solução é executada pelo sistema e os resultados são inseridos na base de casos. Esse tipo de execução envolve, contudo, um alto risco, pois delega muita responsabilidade ao sistema.

Um modo intermediário é a execução da solução proposta de modo automático com o controle do usuário. Nessa modalidade, o usuário pode permitir ou proibir a execução de uma solução que é sugerida pelo sistema, que a executa automaticamente se for autorizado.

Após uma solução ter sido avaliada, se o seu resultado for satisfatório, a experiência que foi obtida durante o processo de resolução do problema corrente deve ser armazenada no sistema. Caso o resultado da avaliação mostre que a solução proposta não produziu um resultado adequado, o caso deve ser reparado. A reparação do caso envolve a detecção dos erros da solução corrente e a recuperação ou geração da explicação para a ocorrência destes erros. Um segundo passo no processo de reparação utiliza, então, as explicações das falhas para modificar a solução corrente de modo que os erros não ocorram mais (SANTOS, 2004).

2.2 TRABALHOS E FERRAMENTAS DE RBC RELACIONADAS

Foram realizadas pesquisas buscando levantar trabalhos com os mesmos objetivos do presente trabalho, ferramentas para o desenvolvimento de aplicações de Raciocínio Baseado em Casos, com foco no ensino da técnica.

O levantamento apontou diversos trabalhos relacionados com ensino-aprendizagem de outros domínios, como educação ou outros temas. Para área de RBC não foi encontrado uma ferramenta com o objetivo de ensino da técnica.

Existem ferramentas com o propósito de desenvolver sistemas de RBC, sendo compatíveis com os dados disponíveis para cada tipo de problema. Hoje encontram-se disponíveis ferramentas comerciais e de software livre. Conforme citado por Watson (1997) e Wangenheim e Wangenheim (2003) as principais são: a) RaBeCa; b) ART Enterprise; c) Induce-It; d) ESTEEM; e) CBR*Tools; f) CBR Works; g) Orenge; h) CBR Shell V2.0; i) jColibri; e j) myCBR.

Nas subseções que seguem tem-se uma breve descrição das características e principais funcionalidades de cada uma destas ferramentas.

2.2.1 RABECA

RaBeCa é um ambiente de desenvolvimento de sistemas de Raciocínio Baseado em Casos híbrido. Ele combina facilidades para apoiar a configuração e a inclusão de diferentes formas de representação de caso, armazenamento e indexação, com uma clara e intuitiva interface gráfica de uso (Figura 3).

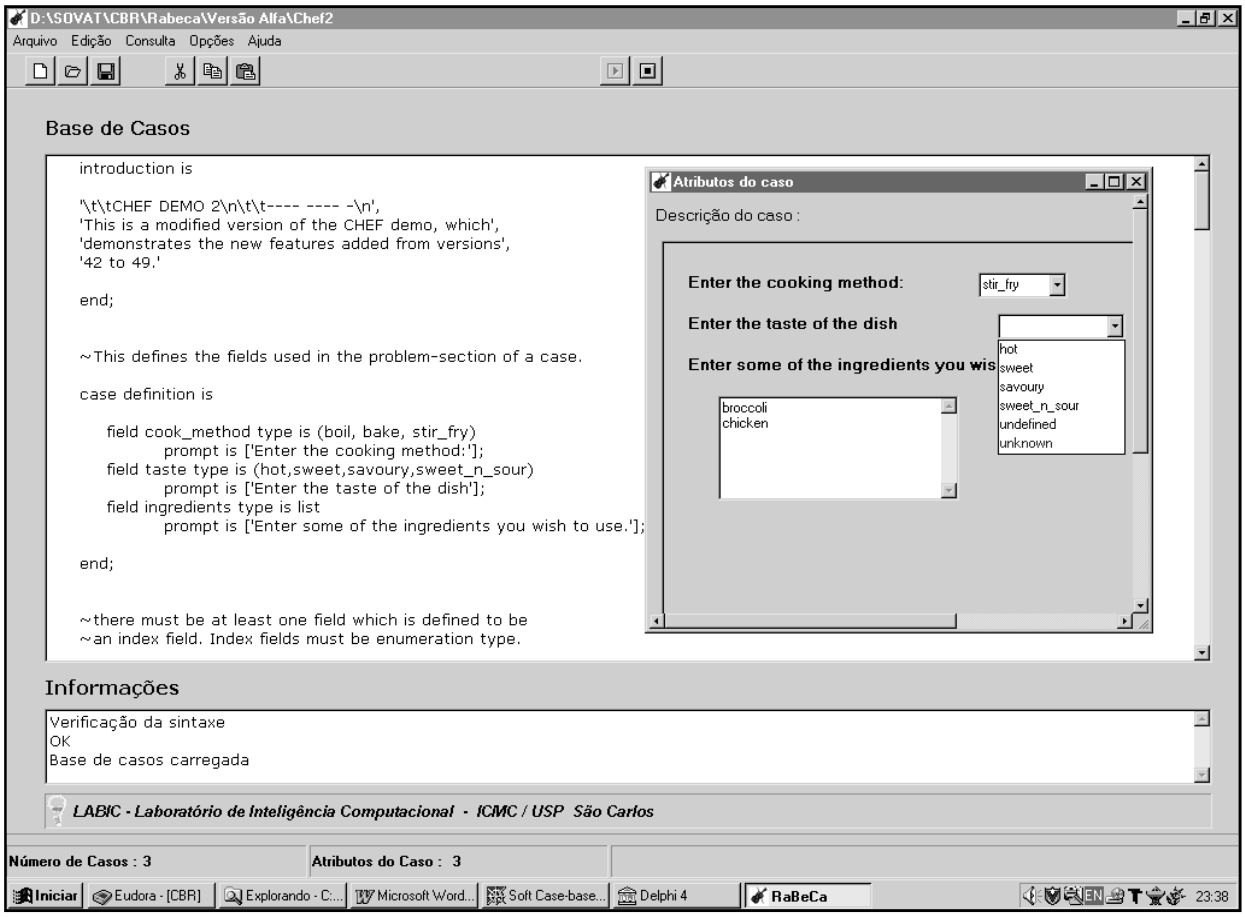


Figura 3. Interface do Ambiente RaBeCa

Fonte: Adaptado Sovat, Aluísio e Carvalho (2001).

RaBeCa é composto por quatro componentes principais: um interpretador, um pré-processador, um indexador e pela interface com o usuário, uma vez que a base de casos é construída dinamicamente. O interpretador aceita a linguagem de *scripts* compatível com CASL, que é uma linguagem desenvolvida pela Universidade do País de Gales, Reino Unido, utilizada para gerar bases de casos para o sistema CASPIAN, porém o sistema foi construído de forma independente.

Os autores citam que a principal motivação para a criação do RabeCa foi a possibilidade de fornecer software aberto o suficiente para permitir a modularidade e até mesmo de elaborar sistemas híbridos. A escolha das possibilidades de oportunidades para hibridação é um dos objetivos da pesquisa. Uma opção de sistema híbrido citado pelo autor é na recuperação das informações da base de casos utilizando-se redes neurais artificiais, onde na falta de algum atributo nos casos armazenados, a rede neural é capaz estimar um valor aproximado para o atributo, melhorando a recuperação dos casos.

2.2.2 ART ENTERPRISE

Segundo Watson (1997), a ferramenta foi criada pela empresa Brightware's, da Califórnia, que desenvolve aplicações de Inteligência Artificial.

ART *Enterprise*[3] funciona como um ambiente de desenvolvimento de aplicações utilizando técnicas de Inteligência Artificial baseadas em casos e baseadas em regras. É recomendado pelo distribuidor para empresas e instituição que desejam implantar aplicativos para otimização e melhoria das práticas empresariais, podendo funcionar com grandes volumes de dados.

As principais características desta ferramenta são: a) Interface XML (*eXtensible Markup Language*) para compartilhamento das informações com outros sistemas; b) Interoperabilidade Java com as funcionalidades da orientação a objetos; c) Balanceamento de carga para utilização com grandes bases de casos e d) Depuração e desempenho permitindo avaliar o desempenho de cada transação realizada.

A ferramenta é comercializada e o site do distribuidor não apresenta a interface da ferramenta para demonstração, com isso não foi possível realizar uma avaliação detalhada da ferramenta.

[3] Disponível em: http://www.mindbox.com/Products/ARTEnterprise.aspx

2.2.3 INDUCE-IT

Induce-It[4] é uma ferramenta para criação de sistemas especialistas baseados em casos em Microsoft Excel. Utilizando a ferramenta o usuário pode localizar os casos mais similares ao problema atual, adaptar o melhor caso a partir das respostas obtidas e realizar a indicação da resposta para o problema atual (INDUCE-IT, 2008).

Se necessário, o novo problema pode ser facilmente armazenado como um novo caso. A similaridade da aplicação pode ser realizada através de funções do ambiente Excel ou utilizando a linguagem C com bibliotecas dinâmicas utilizando o menu personalizado (INDUCE-IT, 2008).

Na Figura 4 é apresentada a interface do ambiente que funciona no Microsoft Excel que pode ser utilizada para edição, gráficos, base de dados, botão, o diálogo, e comandos macro.

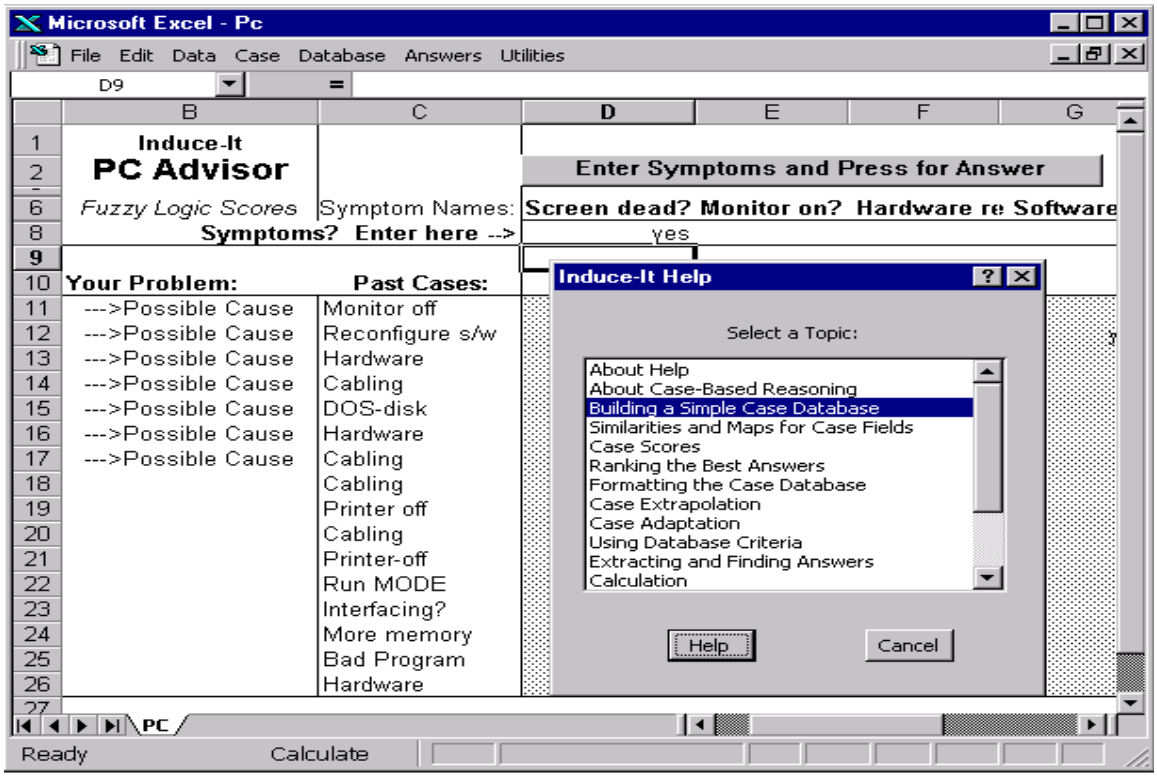


Figura 4. Interface do Ambiente Induce-It

Fonte: Adaptado de Induce-It (2008).

[4] Disponível em: http://www.inductive.com/softcase.htm

2.2.4 ESTEEM

ESTEEM[5] é um software comercial mantido pela empresa Intellicorp's Kappa-PC, que possibilita o usuário criar aplicações utilizando RBC sem precisar desenvolver nenhuma linha de codificação. Esteem disponibiliza a interface para definição da estrutura, índices e adaptação dos casos, como também a interface final com o usuário, conforme apresentado na Figura 5 (ESTEEM, 2008).

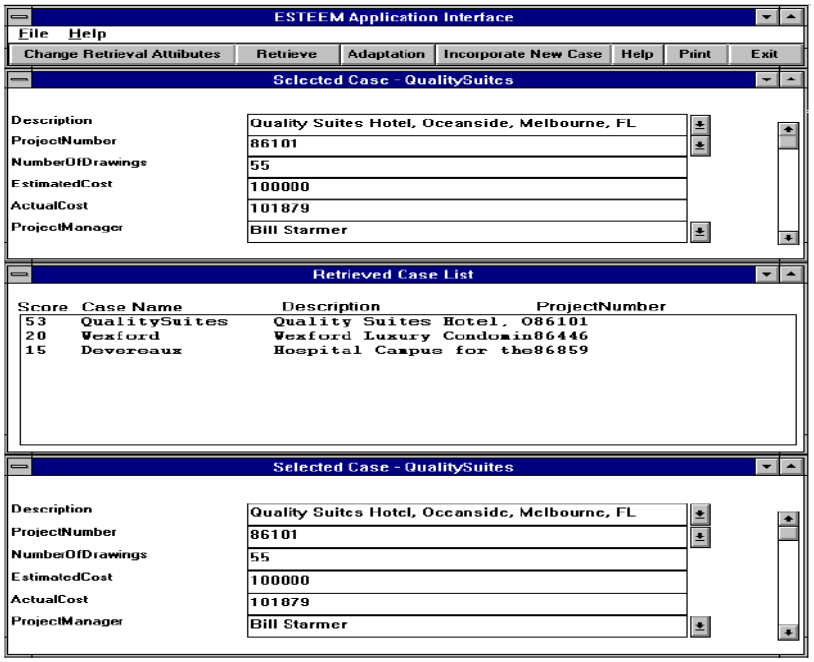


Figura 5. Interface do Ambiente ESTEEM.
Fonte: Adaptado de ESTEEM (2008).

Esteem oferece vários tipos de opções para combinações, dependendo do tipo da característica. Para predefinir as opções de combinações, o desenvolvedor pode também usar a inferência de combinações onde regras específicas são permitidas para definir a similaridade entre as características. A definição da similaridade também permite o desenvolvedor determinar pesos relativos para selecionar as características. A pré-seleção adotada pelo sistema determina o peso para qualquer característica selecionada. A ferramenta fornece duas formas automáticas de generalização de pesos. Uma é pelo algoritmo ID3 e a outra usa Algoritmo Gradiente Descendente (GDA).

[5] Disponível em: http://www.stottlerhenke.com/solutions/decision_support/esteem.htm

2.2.5 CBR*TOOLS

CBR*Tools[6] é um *framework* orientado à objetos criado para o desenvolvimento de sistema de Raciocínio Baseado em Casos. Ele foi desenvolvido utilizando a linguagem de programação Java e utiliza como base o software Rational Rose (Figura 6), sendo este necessário para sua utilização (CBR*TOOLS, 2008).

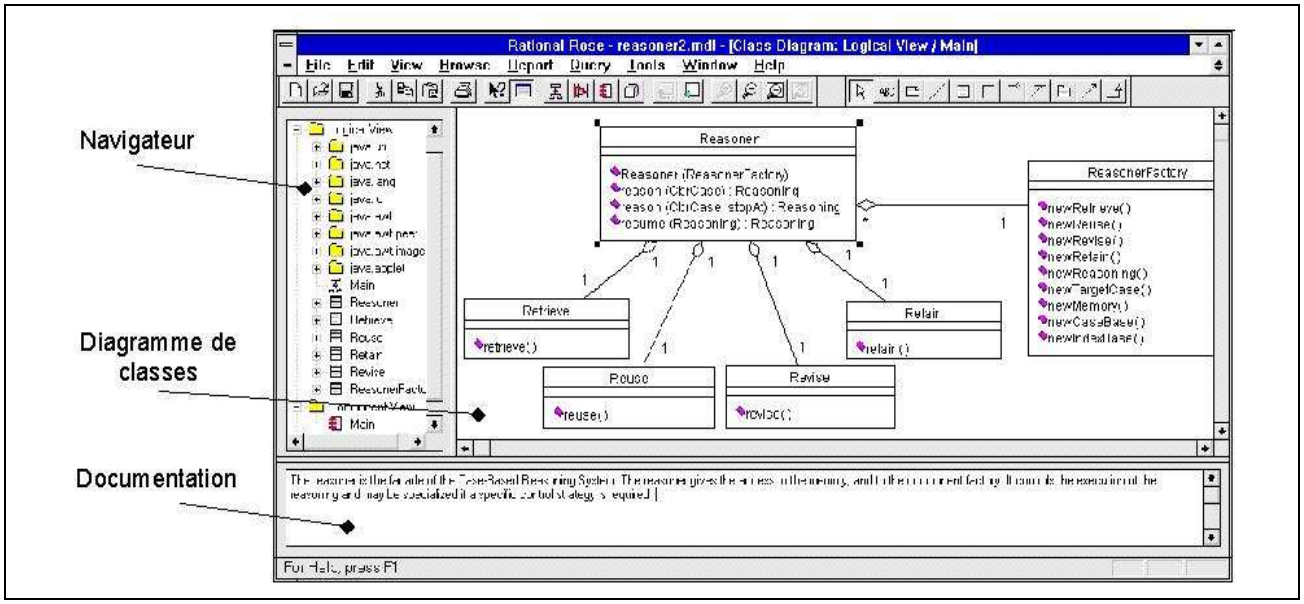


Figura 6. Interface do Ambiente CBR*Tools.
Fonte: Adaptado de CBR*Tools (2008).

Ele fornece um conjunto de classes abstratas para modelagem dos conceitos necessários ao desenvolvimento de sistemas RBC: base de casos, casos, índices, métricas de similaridade e o controle da aplicação. Também fornece classes que implementam os métodos de similaridade tradicionais como Vizinho mais Próximo entre outros.

A ferramenta também possibilita a reutilização e os casos são indexados através de um esquema de situações comportamentais, proposto para ordenar os casos, com base nos comportamentos relevantes.

O software foi desenvolvido pelo Instituto INRIA (Institut National de Recherche em Informatique em Automatique) da França.

---

[6] Disponível em: http://www-sop.inria.fr/axis/cbrtools/manual/

2.2.6 CBR WORKS

CBR-Works[7] é um pacote que está voltado para o desenvolvimento de aplicações dentro do enfoque de sistemas de Raciocínio Baseados em Casos, objetivando a resolução de problemas e aprendizagem com base em experiências passadas. O CBR-Works atende ao processo cíclico, integrado de resolução de um problema, aprendizagem desta experiência e resolução de um novo problema, e assim, sucessivamente (CBR-WORKS, 2008). A interface da ferramenta é apresentada na Figura 7.

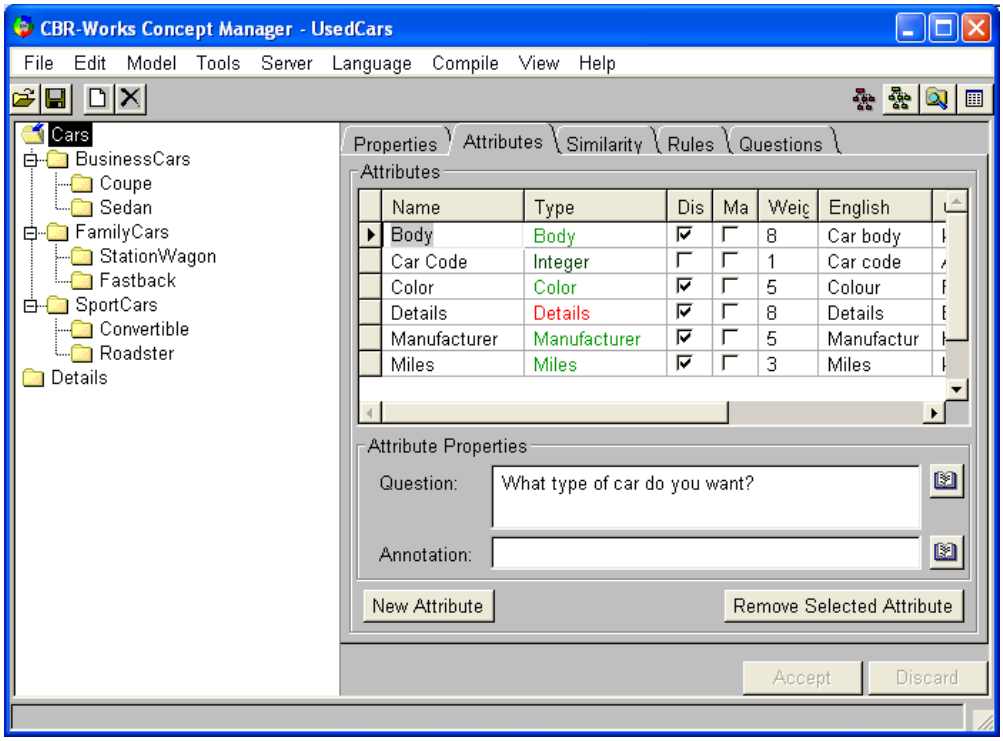


Figura 7. Interface da ferramenta CBR Works.
Fonte: Adaptado de CBR-Works (2008).

A ferramenta comporta tanto o desenvolvimento de uma aplicação nova como o desenvolvimento a partir de um banco de dados já existente, possibilitando assim o desenvolvimento de sistemas com maior rapidez e sem preocupação com elaboração de algoritmos.

CBR-Works foi desenvolvido pela Universidade de Kaiserslautern da Alemanha e pela empresa Tecinno GmbH que comercializa a ferramenta.

---

[7] Disponível em: http://www.tecinno.com/

2.2.7 ORENGE

Orenge (Open Retrieval ENGinE) estabelece um ambiente para desenvolvimento de sistemas de RBC utilizando a linguagem JAVA. Sua principal área de aplicação é servir para o desenvolvimento de aplicações de RBC na área de Raciocínio Baseado em Casos textual, servindo principalmente para desenvolver aplicações comerciais e industriais que lidam com grandes quantidades de texto. Orenge é uma infra-estrutura para o desenvolvimento de máquinas de busca multiconteúdo inteligentes para a gestão de conhecimento em JAVA e XML. Existe tanto para ambientes Linux como para Microsoft Windows.

Segundo Wangenheim e Wangenheim (2003), Orenge foi desenvolvido pelo mesmo grupo de pesquisadores chefiado por Stefan Wess, que desenvolveu CBR-Works, tendo também sido desenvolvido na empresa tecInno/empolis Knowledge Management. Ao contrário de CBR-Works, Orenge é uma ferramenta exclusiva para o desenvolvedor profissional, e não possui uma interface gráfica de usuário. É constituída por uma coleção de ferramentas de linha de comando, que devem ser usadas em conjunto com o editor de programas de escolha do desenvolvedor, no qual os modelos de casos, medidas de similaridade etc., são codificados seguindo a sintaxe definida por Orenge.

Wangenheim e Wangenheim (2003) citam que Orenge inicia uma nova geração de ferramentas de RBC: a das ferramentas complexas e maduras para o desenvolvimento de aplicações comerciais. Ao contrário de CBR-Works, Orenge é uma ferramenta complexa, voltada ao profissional, e não possui o enfoque acadêmico que CBR-Works ainda possuía.

2.2.8 CBR SHELL V2.0

CBR Shell[8] é uma ferramenta do grupo Artificial Intelligence Applications Institute School of Informatics da universidade de Edinburgh. A primeira versão da ferramenta (Java CBR Shell version 1.0) foi desenvolvida em Java e não disponibilizava uma interface para elaboração dos sistemas RBC, porém era possível utilizar a ferramenta através de uma Applet Java em um navegador Web (CBR SHELL, 2008).

---

[8] Disponível em: http://www.aiai.ed.ac.uk/project/cbr/cbrtools.html

A segunda versão (CBR Shell V2.0) foi desenvolvida em Visual Basic e está disponível para uso acadêmico somente para plataforma Windows, onde inicialmente é apresentada como uma barra de ferramentas onde é possível definir as características do sistema. A maior limitação encontrada foi o uso de arquivos textos para armazenar as informações dos casos e também sua interface não apresenta uma seqüência lógica de passos para definição do sistema RBC. A Figura 8 mostra a interface do ambiente (CBR SHELL, 2008).

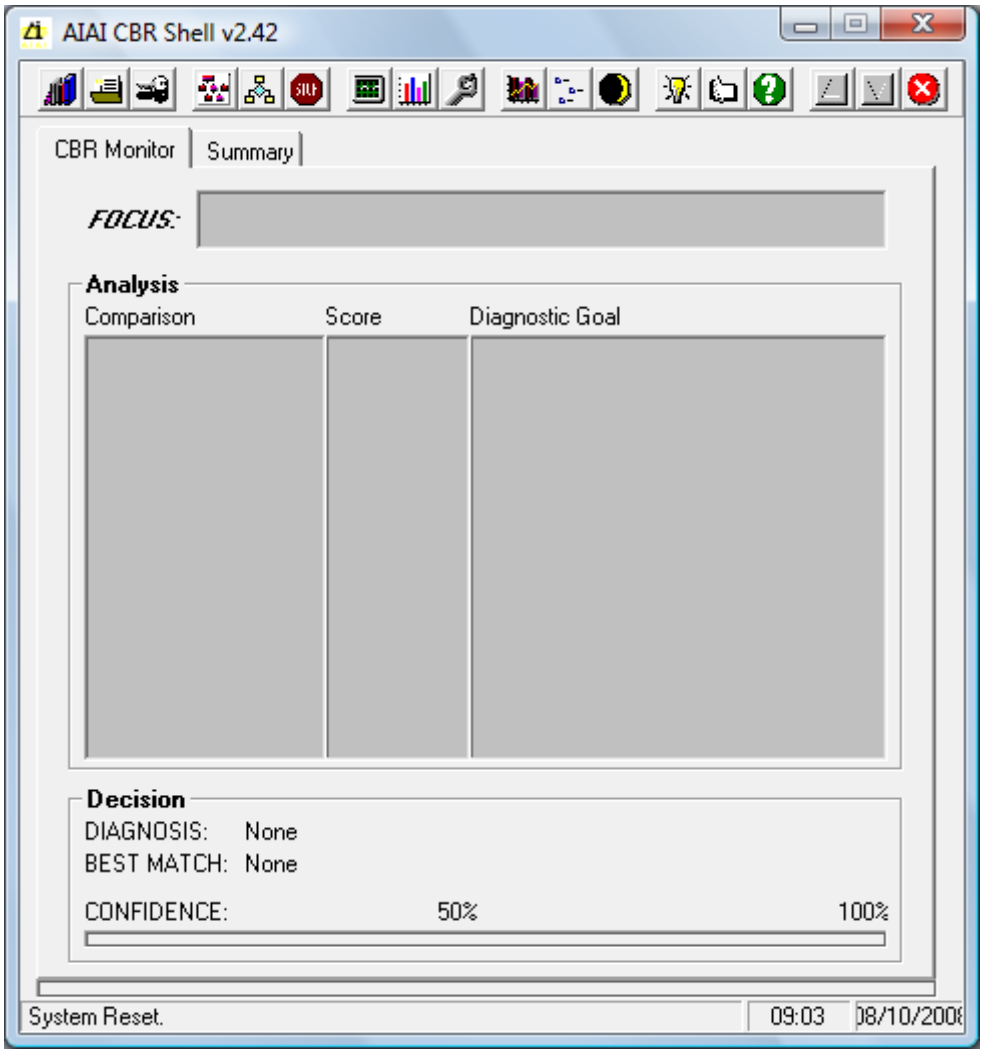


Figura 8. Interface da ferramenta CBR Shell V2.0.

Fonte: Adaptado de CBR Shell (2008).

2.2.9 MYCBR

MyCBR[9] é uma ferramenta Open Source desenvolvida pelo Centro de Pesquisa Alemão de Inteligência Artificial (DFKI). A principal motivação para a implementação myCBR foi a necessidade de uma ferramenta fácil de manusear para construção de protótipos de aplicações RBC, tanto para pesquisa como para pequenos projetos industriais, com o mínimo esforço. A idéia principal da criação da ferramenta veio da antiga ferramenta CBR-Works, que foi descontinuada (MYCBR, 2008). Na Figura 9 é apresentada a interface da ferramenta.

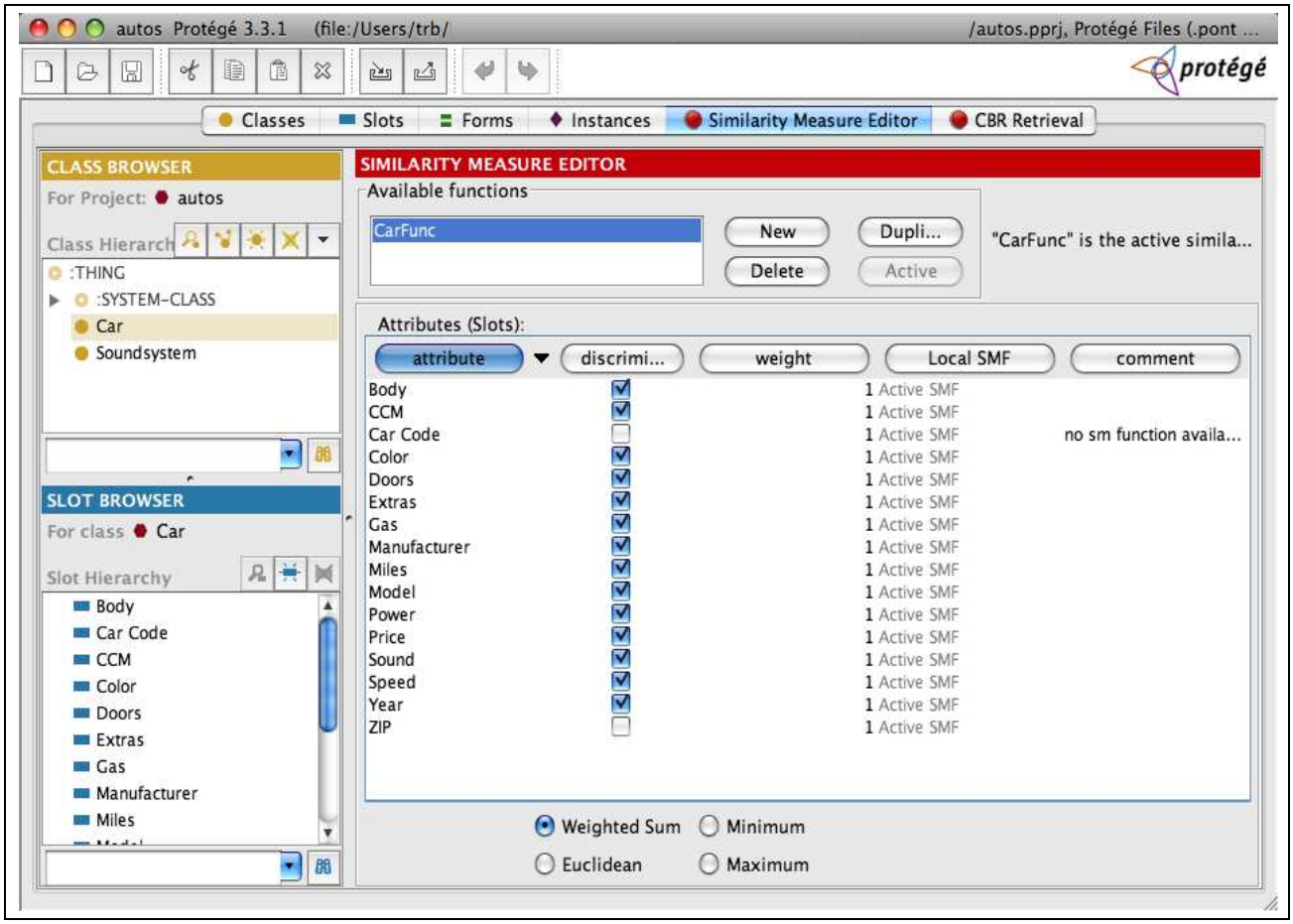


Figura 9. Interface da ferramenta myCBR.

Fonte: Adaptado de myCBR (2008).

[9] Disponível em: http://mycbr-project.net/

A versão atual do myCBR tem foco na similaridade baseada na etapa de recuperação do ciclo do RBC, visto que é a principal funcionalidade das aplicações.

O principal objetivo do myCBR é minimizar o esforço para construir aplicações RBC que requerem alta intensidade cognitiva nas medidas de similaridade. Por isso, ele fornece uma boa interface gráfica para modelar vários tipos de atributos específicos e medidas de similaridade, possibilitando avaliar a qualidade final da recuperação das informações. A fim de reduzir também o esforço da etapa anterior de definição de uma representação de casos adequada, ele utiliza ferramentas para gerar automaticamente o processo de representação de dados brutos.

Uma das deficiências constatadas foi que a ferramenta utiliza apenas arquivos XML para armazenar as informações, não dando suporte para outros tipos de armazenamento.

2.2.10 JCOLIBRI

jColibri[10] é um *framework* orientado a objetos que facilita a construção de sistemas de Raciocínio Baseado em Casos (RBC). Foi projetado pensando-se em conseguir uma plataforma de desenvolvimento de sistemas RBC que servisse de referência na comunidade científica.

Na arquitetura de jColibri foram utilizadas técnicas de desenvolvimento no campo da Engenharia de Software para promover a reutilização do software, e as idéias propostas pela metodologia KADS (*Knowledge-Based System Development*), como afirmam Schreiber, Wielinga e Breuker (2003 *apud* RUIZ-GRANADOS, 2005), basicamente consistem em separar os métodos que obtêm a solução de um problema do modelo de domínio que descreve o conhecimento específico de cada aplicação. jColibri foi desenvolvido em Java para facilitar sua utilização em distintos ambientes utilizando tecnologias amplamente aceitas como XML e JBDC.

Portanto, jColibri torna-se uma ferramenta que permite construir aplicações RBC concretas de forma visual e guiada. Embora no momento a ferramenta não consiga gerar uma aplicação completa, serve para criar o esqueleto de uma aplicação. Este tipo de ferramenta ajuda a aprendizagem inicial para qualquer um que esteja interessado em aprender a utilizar o *framework*.

---

[10] Disponível em : http://gaia.fdi.ucm.es/projects/jcolibri/

jColibri é uma evolução de um sistema anteriormente denominado Colibri, afirma Diaz-Agudo (2002 *apud* RUIZ-GRANADOS, 2005). Este sistema se concentra em RBC com conhecimento intensivo (K-I RBC), os quais se caracterizam por utilizar conhecimento adicional sobre o domínio em que operam e, em teoria, deveriam melhorar sua eficiência (RUIZ-GRANADOS, 2005).

O sistema Colibri utilizava uma ontologia de concepções em sistemas RBC denominada CBROnto, segundo informações de Diaz-Agudo e Gonzáles-Calero (2002 *apud* RUIZ-GRANADOS, 2005). Essa ontologia define um vocabulário comum sobre o que existe desenvolvido na biblioteca de métodos de resolução de problemas (*Problem Solving Methods* ou PSM). Essa biblioteca permite resolver tarefas comuns em sistemas RBC sem a necessidade de utilizar o conhecimento específico do domínio, o que propicia a sua reutilização.

Colibri foi desenvolvido em LISP (*List Processing Language*) e utilizava LOOM (*Ontology Markup Language*) como tecnologia para representação do conhecimento. O sistema foi utilizado durante alguns anos, porém era pouco utilizado fora do grupo de pesquisa onde foi desenvolvido. jColibri herdou muitas idéias originais deste sistema, tentando se transformar em uma ferramenta utilizável pela comunidade.

O jColibri é um *framework* que foi pensado e desenvolvido em forma de tarefas, sendo cada uma delas associada a uma técnica de RBC. As principais são:

- recuperação dos casos mais similares (*CBR Retrieve Task*);
- reutilização do caso para resolver o problema (*CBR Reuse Task*);
- revisão do caso da solução proposta (*CBR Revise Task*); e
- aprendizado com a experiência (*CBR Retein Task*).

jColibri possui duas versões disponíveis aos desenvolvedor. A versão anterior 1.1 disponibiliza um ambiente não muito usual para elaborar o sistema, conforme apresentado na Figura 10 , onde o usuário pode utilizar os tipos de dados e métricas pré-definidas no ambiente ou criar suas próprias utilizando a linguagem de programação Java. A versão 2 não possui interface gráfica para elaboração e testes do sistema RBC, disponibilizando somente a API para concepção de sistemas.

As tarefas disponibilizadas ao desenvolver uma aplicação são armazenadas em arquivos no formato xml (*tasks*.xml). Para acrescentar novas tarefas é necessário acrescentá-las a esse arquivo. Os métodos também são definidos em um arquivo no formato xml (*methods*.xml) por serem uma representação mais complexa.

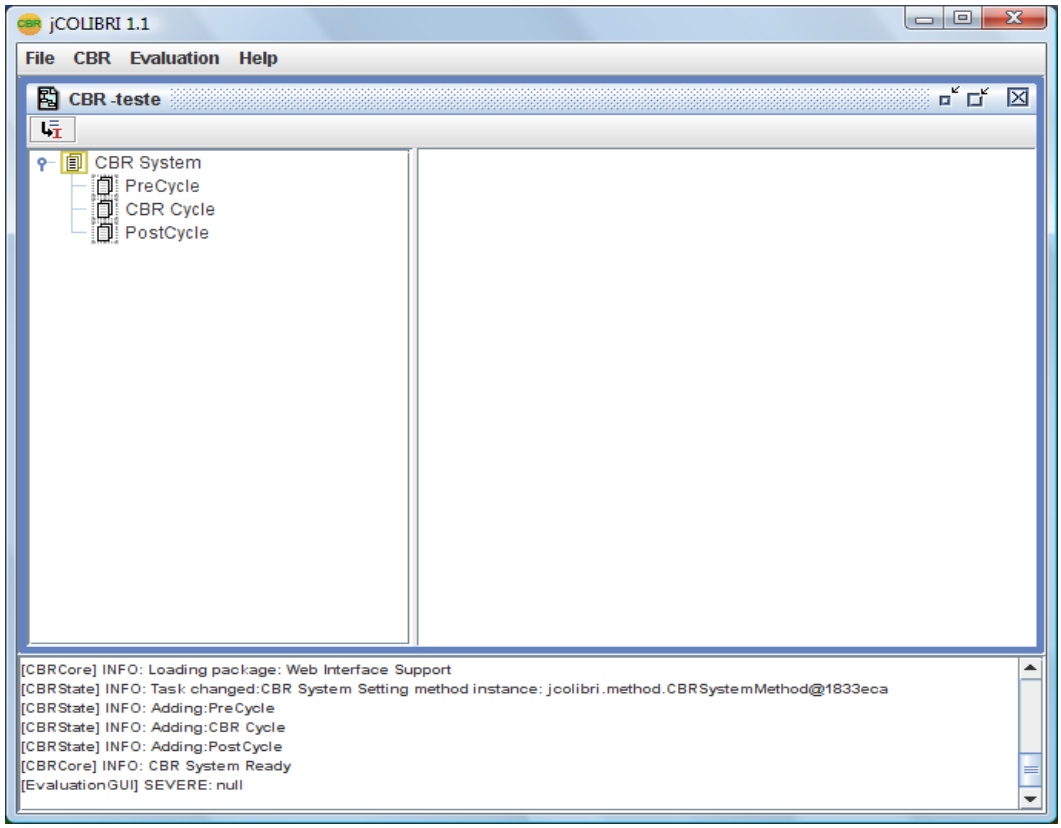


Figura 10. Interface da ferramenta jColibri 1.1.

Fonte: Adaptado de GAIA (2008).

Em jColibri a gerência da base de casos se divide em duas: o mecanismo de persistência utilizado para alcançar os casos e a organização em memória dos casos recuperados. Os casos são representados de maneira muito generalizada. Um caso se modela como um indivíduo que se relaciona com outros indivíduos (os atributos do caso). Essa representação permite construir estruturas arbitrariamente complexas e tratá-las de maneira uniforme (RUIZ-GRANADOS, 2005).

2.3 COMPARATIVO ENTRE AS FERRAMENTAS

Para a realização da análise comparativa entre as ferramentas foram levantadas as características apresentadas pelos fabricantes, e quando disponível, foram instaladas as ferramentas e realizados testes para verificação das características.

Como formas de comparação foram definidos parâmetros de maneira à apresentar as principais características e funcionalidades disponibilizadas pelas ferramentas. Os parâmetros selecionados foram:

- Licença: definindo a forma com que a ferramenta pode ser utilizada, descrevendo se é paga (comercial), ou é disponibilizada gratuitamente para uso não comercial ou direito total sobre a ferramenta, disponibilizando seu código fonte, podendo desta forma modificá-la ou vendê-la se desejar;

- Armazenamento: forma como podem ser armazenadas as informações do sistema RBC (base de casos), descrevendo as tecnologias disponibilizadas como, por exemplo, Arquivos Texto, Arquivos XML (eXtensible Markup Language), Banco de Dados, etc;

- Linguagem: linguagem de programação em que a ferramenta foi desenvolvida e a linguagem utilizada para implementar novas funcionalidades no sistema RBC;

- Geração de Interface: possibilidade da ferramenta produzir automaticamente uma interface final para o usuário utilizar o sistema RBC, sem a necessidade de instalar a ferramenta de desenvolvimento;

- Tipos de Métricas: tipos de métricas de similaridade implementadas pela ferramenta;

- Tutorial: disponibilidade de tutoriais ou ajudas que descrevam como desenvolver um sistema utilizando a ferramenta;

- Ergonomia e Usabilidade: facilidade de uso e compreensão das funcionalidades disponíveis na ferramenta, avaliando a capacidade de um usuário leigo (sem conhecimento da ferramenta) em desenvolver um sistema RBC através da interface.

Utilizou-se na avaliação os quantificadores de qualidade: Ruim, Regular, Bom, Ótimo;

- Interoperabilidade: capacidade da ferramenta se comunicar de forma transparente com outro sistema ou ferramenta, como no caso do armazenamento em banco de dados;

- Plataforma: plataforma ou sistema operacional em que a ferramenta foi desenvolvida e possibilita o uso; e

- Atualizações: respostas fornecidas pelos fabricantes/desenvolvedores quanto à disponibilização de novas funcionalidades, correção de problemas identificados pelos usuários e dúvidas através de fóruns ou emails de contato.

Avaliando cada um destes parâmetros obteve-se ao fim da análise um quadro comparativo, conforme apresentado no Quadro 1 e Quadro 2. Em algumas situações não foi possível obter as informações sobre determinadas características da ferramenta pois não foi possível realizar testes e/ou a descrição fornecida pelo fabricante/desenvolvedor não detalhava o suficiente, e nesta situação colocou-se como não disponível.

Quadro 1. Comparativo entre as ferramentas 1

	Licença	Armazenamento	Linguagem	Geração de Interface	Tipos de Métricas
RaBeCa	Proprietário	Não disponível	Desenvolvido em Delphi Possibilita o uso da linguagem de scripts compatível c/ CASL	Não	Vizinho mais próximo Distância Euclidiana
ART Enterprise	Comercial	Banco de dados Documentos	Desenvolvido em Java	Não	Não disponível
Induce-It	Comercial	Planilha MS Excel	Desenv. em MS Excel Macros do MS Excel	Não	Não disponível
ESTEEM	Comercial	Banco de Dados Arquivos Texto ASCII	Não disponível	Não	Contagem Características Vizinho mais próximo
CBR*Tools	Comercial	Banco de Dados	Java	Não	Vizinho mais próximo
CBR Works	Comercial	Banco de Dados	Não divulgado Permite utilizar linguagem de scripts própria para similaridade	Não	Média (Average) Mínimo Máximo Euclidiana
Orenge	Comercial	Não disponível	Não disponível	Não	Vizinho mais próximo
CBR Shell V2.0	Acadêmico	Arquivos Texto ASCII	Não divulgado Não permite implementar novas funcionalidades	Não	Média (Average)
jColibri 1.1	Software Livre	Banco de dados Arquivos Texto	Java	Sim	Média (Average)
myCBR	Software Livre	Arquivos XML	Java	Não	Soma dos pesos Mínimo Máximo Euclidiana

Quadro 2. Comparativo entre as ferramentas 2

	Tutorial	Ergonomia e Usabilidade	Interoperabilidade	Plataforma	Atualizações
RaBeCa	Não	Interface Textual Ruim	Não disponível	Windows	Não disponível
ART Enterprise	Não Disponível	Não Disponível	Com ferramentas Java	PC e Mainframe	Não disponível
Induce-It	Não	Interface do MS Excel Ruim	Não	Windows	Não disponível
ESTEEM	Não Disponível	Regular	Não	Windows	Descontinuado
CBR*Tools	Não	Rational Rose Ruim	Com ferramentas Java	Windows	Descontinuado
CBR Works	Ajuda Simples	Boa	Não	Windows	Descontinuado
Orenge	Não disponível	Não disponível	Não disponível	Windows	Não disponível
CBR Shell V2.0	Não	Ruim	Não	Windows	Última atualização em Fevereiro/2004
jColibri 1.1	Sim Diretamente no site do fabricante, Não interativo	Regular	Sim	Sistemas Compatíveis com Java	Última atualização em Janeiro/2008
myCBR	Não	Boa	Sim	Sistemas Compatíveis com Java	Ultima atualização em Outubro/2008

Com o comparativo entre as ferramentas foi possível constatar que muitas das ferramentas foram descontinuadas e não possuem mais suporte aos usuários. As principais ferramentas existentes para o desenvolvimento de sistemas RBC que ainda possuem suporte são jColibri e myCBR.

A ferramenta myCBR está em constante desenvolvimento e utiliza outra ferramenta de Modelagem do Conhecimento e Ontologias chamada Protégé, que por incompatibilidade entre as versões disponibilizadas pelos fabricantes não foi possível realizar testes com esta ferramenta.

A versão avaliada da ferramenta jColibri foi a 1.1, que possui uma ferramenta de autoria onde o usuário pode realizar a modelagem do sistema. A ferramenta jColibri possui uma versão 2.0 que não disponibiliza a ferramenta de autoria, porém foram corrigidos diversos problemas avaliados na versão anterior, onde o desenvolvimento de sistemas RBC nesta ferramenta obriga os desenvolvedores a configurarem todo o sistema através dos arquivos XML de configuração.

Através desta análise foi possível avaliar as principais características e funcionalidades já disponibilizadas pelas ferramentas, observando a limitação do *framework* jColibri 2.1 de não possuir uma ferramenta visual, porém sua flexibilidade chama a atenção. Desta forma foi realizada uma análise detalhada da arquitetura e do funcionamento deste *framework.*

2.4 ARQUITETURA DO JCOLIBRI

jColibri é um *framework* orientado a objetos *Open Source* que facilita a construção de sistemas de Raciocínio Baseado em Casos (RBC). Foi projetado pensando-se em conseguir uma plataforma de desenvolvimento de sistemas RBC que servisse de referência na comunidade científica.

Segundo Johnson e Foot (1988 *apud* RECIO-GARCIA, DIAZ-AGUDO & GONZÁLES-CALERO, 2008) “um *framework* é um conjunto de classes que incorpora uma concepção abstrata para solução de uma família de problemas relacionados”, ou seja, um projeto e implementação parcial para uma aplicação em um determinado domínio de problema.

Os *framework* podem ser classificados em dois grupos distintos: caixa branca e caixa preta. Um *framework* caixa branca baseiam-se nos mecanismos de herança e reuso presentes principalmente na orientação a objetos. Um *framwork* caixa preta é baseado em componentes de

software, onde os recursos existentes são reutilizados e estendidos por meio da definição de um componente adequado a uma interface específica. (GAMMA *et al*, 1994)

A primeira versão de jColibri é fechado como um *framework* caixa-preta, composto por construtor visual e carece de uma estrutura caixa-branca. A nova versão de jColibri foi então remodelada sua arquitetura e constituída como um *framework* caixa-branca orientado a programadores, e um *framework* caixa preta com uma camada orientada a *designers* através de um construtor. Segundo Recio-Garcia, Diaz-Agudo e Gonzáles-Calero (2008) a principal idéia do novo *design* consiste em separar as classes núcleo (*core*) e a interface do usuário, através de duas camadas conforme apresentado na Figura 11.

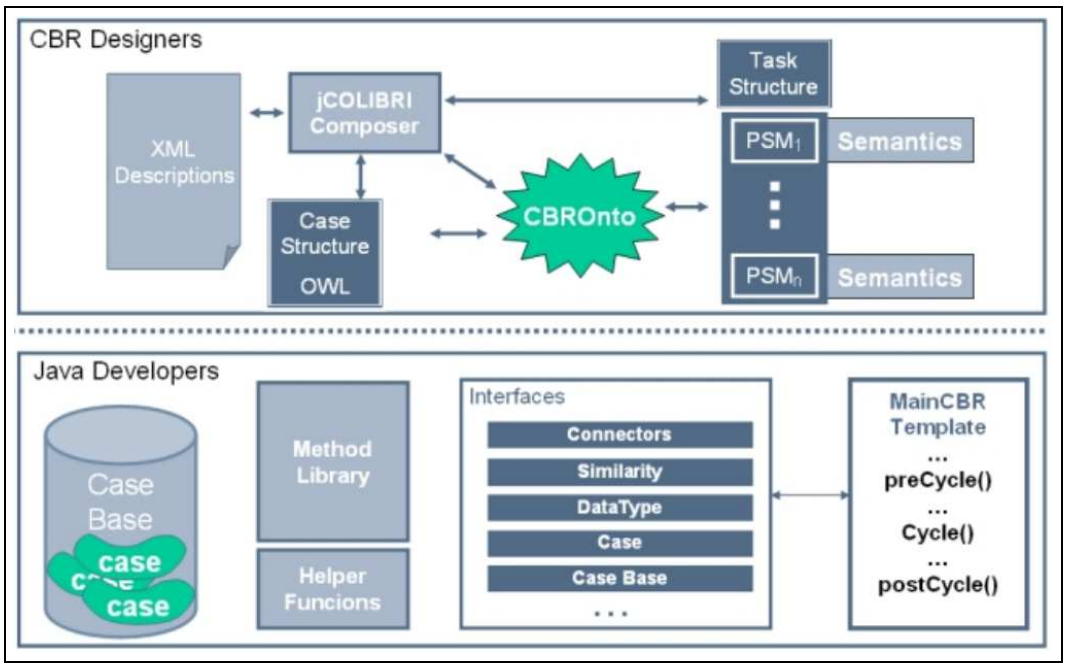


Figura 11. Arquitetura de jColibri2 em duas camadas

Fonte: Adaptado de Recio-Garcia, Diaz-Agudo e Gonzáles-Calero (2008)

Conforme descrito por Recio-Garcia, Diaz-Agudo e Gonzáles-Calero (2008):

> "A camada inferior (Java Developers) contém os componentes básicos do framework com interfaces limpas e bem definidas. Esta camada não contém qualquer ferramenta gráfica para desenvolver aplicações RBC, sendo simplesmente um framework caixa-branca orientado a objetos para ser utilizado por programadores. A camada superior (CBR Designers) contém descritores semânticos dos componentes e várias ferramentas para auxiliar usuários no desenvolvimento de aplicações RBC (caixa-preta com framework de construção visual)"

A camada inferior possui novas funcionalidades onde foram resolvidos muitos problemas identificados na primeira versão. A vantagem desta nova versão são as novas possibilidades oferecidas para integrar novas funcionalidades da linguagem Java, como a representação dos casos através de Java Beans, sendo utilizado para manipular a persistência dos casos o pacote Hibernate[11].

O Hibernate é uma ferramenta de mapeamento objeto/relacional para Java. Ela transforma os dados tabulares de um banco de dados em um grafo de objetos definido pelo desenvolvedor. Usando o Hibernate, o desenvolvedor evita escrever muito do código de acesso a banco de dados e de SQL que ele escreveria não usando a ferramenta, acelerando a velocidade do seu desenvolvimento de uma forma fantástica.

Java Beans e Hibernate são tecnologias utilizadas na plataforma Java Enterprise Edition e são orientadas a aplicações de negócio. Usando estas tecnologias em jColibri2, Recio-Garcia, Diaz-Agudo e Gonzáles-Calero (2008), garantem a integração de aplicações comerciais com RBC usando o *framework.*

O *framework* jColibri2 implementa todo o ciclo RBC, é disponibilizado como software livre e está disponível para *download* no *web site*[12] da ferramenta ou através do repositório *SourceForge*[13]. Por ser desenvolvido em Java, é necessário previamente possuir instalado o Kit de desenvolvimento Java (Java SDK[14]). O pacote jColibri2 disponibiliza juntamente com o código fonte do *framework* um conjunto de 30 exemplos que podem ser visualizados através de uma interface gráfica, onde é possível avaliar o funcionamento e desempenho do *framework*, também disponibilizando toda a documentação dos exemplos.

Para criação de sistema de RBC utilizando jColibri2, torna-se necessário a definição da estrutura dos casos. Os componentes do caso podem ser divididos em quatro componentes: descrição do problema, solução do problema, resultado da aplicação da solução e justificativa para a solução. Cada um destes componentes pode ser representado por um ou mais atributos e para isso jColibri disponibiliza uma estrutura de classes (orientada a objetos), que deve ser utilizada, conforme apresentada na Figura 12.

---

[11] http://www.hibernate.org/
[12] http://gaia.fdi.ucm.es/projects/jcolibri/jcolibri2/index.html
[13] http://sourceforge.net/projects/jcolibri-cbr/
[14] http://java.sun.com/

Na estrutura de representação dos casos são definidas quatro classes principais, que armazenam a descrição do problema (CBRQuery), a solução do problema com o resultado e a justificativa (CBRCase) e as classes responsáveis por armazenar os atributos do caso (CaseComponent e Attribute).

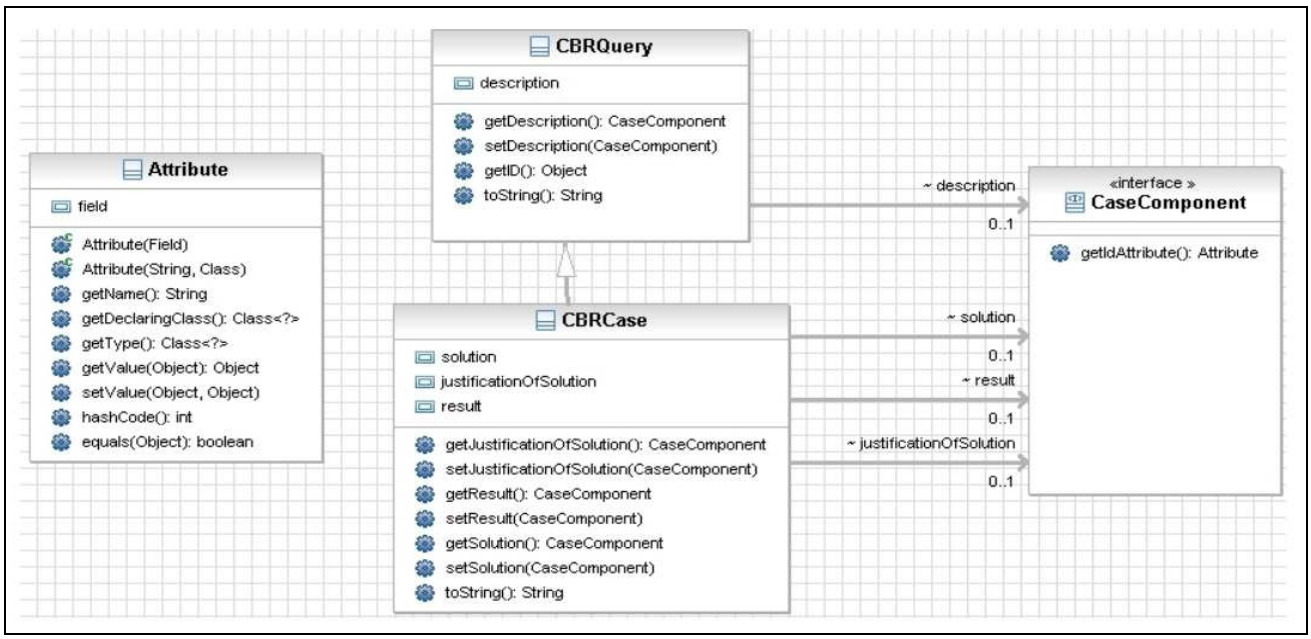


Figura 12. Representação dos Casos (Diagrama UML)
Fonte: Adaptado de Recio-Garcia, Diaz-Agudo e Gonzáles-Calero (2008).

Após a definição da estrutura dos casos, o próximo passo é a definição da forma como será realizado o acesso e a persistência dos dados. Em sistemas RBC o acesso a base de casos armazenada deve ser realizado de forma eficiente, pois isso pode se tornar um problema cada vez mais relevante quando a dimensão da base de casos aumenta. jColibri divide este processo de organização de duas maneiras distintas, embora relacionadas: mecanismo de persistência e a organização em memória.

A Figura 13 ilustra a organização da base de casos, onde toda a persistência é realizada através de conectores, que realizam o acesso, recuperação e gravação de casos no meio físico,

persistindo através de banco de dados (utilizando biblioteca Hibernate), arquivo texto (arquivos textuais) e ontologias (baseado em ontologia utilizando o OntoBridge[15]).

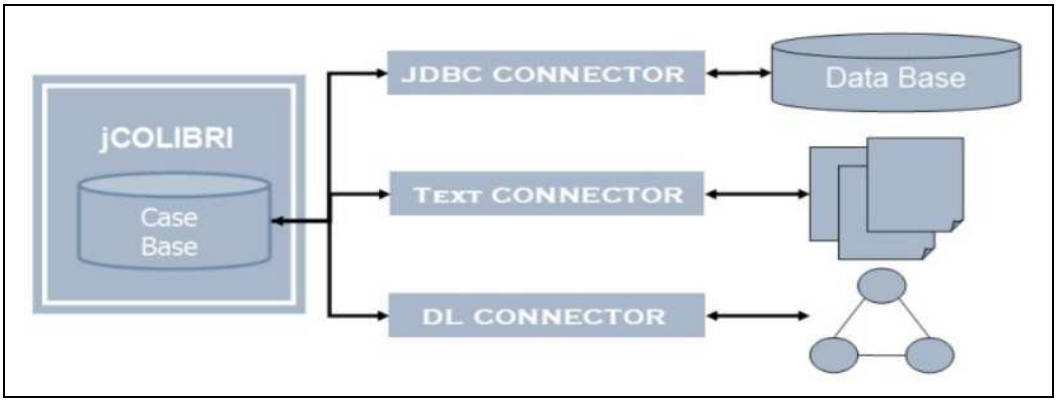


Figura 13. Organização da Base de Casos (Persistência)
Fonte: Adaptado de Recio-Garcia, Diaz-Agudo e Gonzáles-Calero (2008)

A organização da base de casos em memória para ser utilizada pelo *framework* é de suma importância no processo de RBC, pois influencia diretamente no desempenho da aplicação. Para isso jColibri implementa três formas possíveis: linear (casos são gravados em uma lista de forma linear), cache linear (casos são gravados somente quando a aplicação é encerrada) e linear indexada (extensão da linear, onde são mantidos índices para os casos e utilizados como Ids ou identificadores).

A implementação de aplicações utilizando o *framework* jColibri é baseada na extensão da interface "jcolibri.cbrapplications.StandardCBRApplication" (Figura 14), que deve ser implementada pela aplicação que está sendo desenvolvida e que define o comportamento do sistema RBC, através de quatro métodos principais:

- configure( ): responsável por realizar as configurações do sistema RBC, definir os conectores e a base de casos.

- preCycle ( ): executado antes do ciclo RBC e tipicamente é responsável por realizar a leitura e organização da base de casos.

[15] OntoBridge (http://gaia.fdi.ucm.es/projects/ontobridge/) é uma biblioteca desenvolvida pelo grupo GAIA que facilita a manipulação de ontologias.

- cycle ( ): executa o ciclo RBC, onde são analisados os casos da base e computada a similaridade e as adaptações definidas, finalizando com a apresentação das informações ao usuário.

- postCycle ( ): executado quanto é encerrada a aplicação, utilizado tipicamente para fechar o conector.

```
package jcolibri.cbraplications;


import jcolibri.cbrcore.CBRCaseBase;
import jcolibri.cbrcore.CBRQuery;
import jcolibri.exception.ExecutionException;


public interface StandardCBRApplication
{
    public void configure() throws ExecutionException;

    public CBRCaseBase preCycle() throws ExecutionException;

    public void cycle(CBRQuery query) throws ExecutionException;

    public void postCycle() throws ExecutionException;
}
```

Figura 14. Interface jcolibri.cbrapplications.StandardCBRApplication
Fonte: Adaptado de GAIA (2008)

Ao executar a aplicação RBC, o método “configure( )” e “preCycle ( )” são responsáveis por definir as configurações e recuperar a base de casos em memória. Na seqüência o método “cycle( )” realizará o ciclo RBC (4’R : recuperação, reuso, revisão e retenção).

A recuperação em jColibri é realizada através da métrica de similaridade global *Nearest Neighbor* ou Vizinho mais próximo, conforme apresentada na Seção 2.1.7.1, utilizando o cálculo da similaridade local de cada atributo do caso, relacionado com seu respectivo peso definido.

Para o cálculo da similaridade local jColibri2 disponibiliza métodos para bases de casos textuais e ontologias (que não serão abordadas na ferramenta proposta) e também outros métodos para os atributos do caso. Entre os métodos disponíveis estão: Símbolos ordenados (Enumerados), igualdade, intervalos entre valores, contagem de características (palavras), *threshold* e Matriz de Similaridade. Cada método é implementado por uma classe do pacote “jcolibri.method.retrieve.NNretrieval.similarity.local”.

Para a etapa de reuso ou adaptação do caso recuperado para a situação atual, jColibri disponibiliza basicamente dois métodos de adptação: DirectAttributeCopyMethod (copia os valores dos atributos do caso anterior para o caso atual, ou seja, adaptação nula) e

NumericDirectProportionMethod (realiza a proporcionalidade numérica direta entre atributos do caso anterior e o atual, apresentando um valor cálculo ao usuário, ou seja, adaptação transformacional).

Na etapa de revisão, onde o caso mais similar à situação atual é revisado pelo usuário para que seja possível armazenar na base de casos para uso futuro, jColibri2 possibilita que o usuário visualize as informações e disponibiliza um método (DefineNewIdsMethod) responsável por gerar um novo identificador para o caso que será armazenado.

Na etapa de retenção, jColibri2 realiza a adição do novo caso a base de casos existente, sendo realizada de maneira transparente, pois a gravação é dependente da forma como foi definida a organização em memória da base de casos (linear, cache linear ou linear indexada).

Na Figura 15 é apresentado um Diagrama de Pacotes que exibe as principais classes do *framework*, mostrando a estrutura que é disponibilizada aos desenvolvedores.

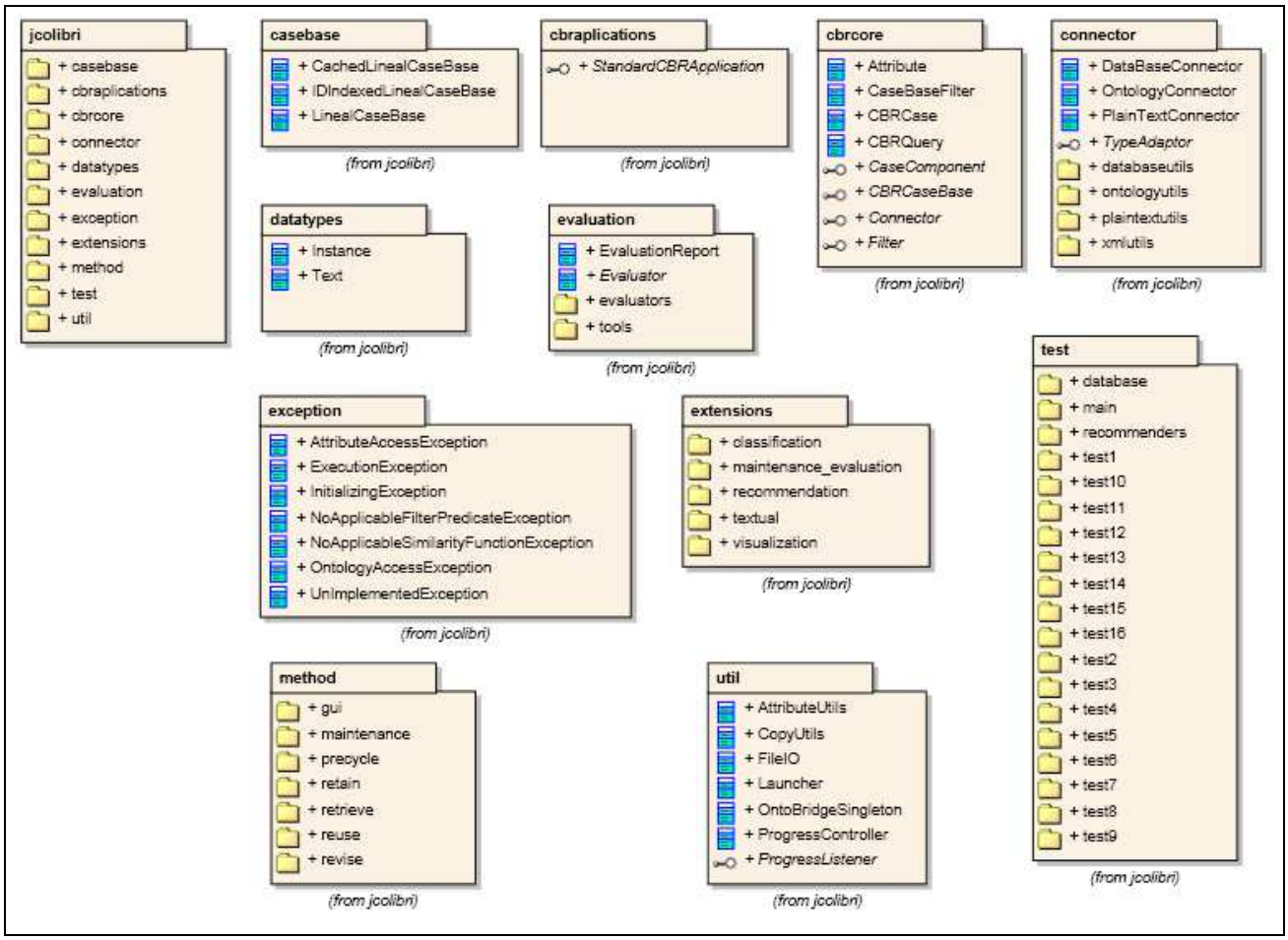


Figura 15. Pacotes do *framework* jColibri2

As principais classes para o desenvolvimento da ferramenta proposta estão incluídas no pacote **casebase, cbcore e method**. O pacote **jcolibri.casebase** incluí as formas de representação dos casos em memória. O pacote **jcolibri.cbcore** incluí as classes responsáveis pela representação dos casos. O pacote **jcolibri.method** inclui os métodos para execução do ciclo RBC.

No desenvolvimento de aplicações RBC utilizando jColibri2, toda a configuração dos conectores da base de casos e da biblioteca Hibernate (responsável pela persistência dos dados) é realizada através de arquivos XML[16]. Além dos arquivos de configuração, em situações onde são utilizados bancos de dados como forma de persistência para base de casos, é necessária a configuração dos arquivos da biblioteca Hibernate que relacionam as tabelas e campos do banco de dados com os atributos das classes criadas para armazenar as informações do caso.

A implementação das etapas do ciclo RBC é realizada utilizando a linguagem de programação Java. Para criação de novas funcionalidades, como novas métricas de similaridade ou outras formas de adaptação dos casos, torna-se necessário a extensão das classes base do *framework* e a implementação destes novos métodos.

A ferramenta que está sendo proposta neste trabalho, busca reduzir a complexidade da configuração destes arquivos, bem como otimizar e reduzir as implementações necessárias para se criar um sistema RBC, possibilitando a fácil integração com diversos sistemas de banco de dados existentes.

[16] XML (eXtensible Markup Language) é uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais, sendo considerada uma linguagem de programação utilizada para facilitar a partilha de dados entre os diferentes sistemas de informação.

2.5 ESTRATÉGIA DE ENSINO

Um sistema computacional é uma das facilidades que a tecnologia da informação e comunicação oferece e que podem ser utilizados como um ambiente de ensino-aprendizagem com a finalidade de possibilitar maior efetividade nas atividades de ensino, de aprendizagem, de avaliação e de comunicação.

A utilização do computador na educação vem demonstrando ser um grande auxílio no processo de ensino-aprendizagem. Um software educacional possui o objetivo de auxiliar o professor no processo de ensino-aprendizagem, fazendo com que o mesmo tenha a seu dispor valiosos recursos para ajudá-lo com seus alunos (HERCULIANI, 2007).

Um software educacional, segundo Lucena (1992 *apud* HERCULIANI, 2007), é todo programa que possa ser usado para algum objetivo educacional, pedagogicamente defensável, por professores e alunos, qualquer que seja a natureza ou finalidade para o qual tenha sido criado.

Neste contexto estão as teorias de aprendizagem que descrevem e possibilitam definir e classificar as estratégias de ensino que podem ser abordadas no processo de ensino aprendizagem. Segundo Mergel (1998) as teorias de aprendizagem são divididas em Comportamentalismo ou Behaviorismo, Cognitivismo e Construtivismo.

Outras teorias explanadas podem ser apresentadas, porém as três teorias citadas são as grandes vertentes, e ao serem analisadas com maior atenção, é possível perceber diversas coincidências em suas idéias e princípios, tornando possível incluir princípios das teorias na elaboração de ambientes de ensino-aprendizadem. Estratégias comportamentalistas podem ser utilizadas para ensinar fatos (definições), as cognitivistas podem ser aplicadas no ensino de processos e princípios (elaboração) e as construtivistas podem auxiliar estudantes no aprendizado contextual (ALLY, 2004).

Autores (JONES, 2002; COOL *et al*, 2003) propõem a adoção de metodologias de ensino em que o aluno possa aprender através da dúvida, do questionamento, deixando de lado o papel de receptor e tornando-se construtor do próprio conhecimento, percebendo sua importância nesse processo de construção e as implicações que seus atos geram nas atividades educativas.

É com esse enfoque metodológico construtivista, que visa a participação do aluno, o desenvolvimento de sua capacidade crítica e de auto-aprendizagem, que a aprendizagem baseada

em problemas (PBL ou Problem-based Learning) vem se tornando aliada para professores do Ensino Fundamental, Médio e Superior (ANDRADE & CAMPOS, 2007).

2.5.1 APRENDIZAGEM BASEADA EM PROBLEMAS (PBL)

A PBL é considerada uma abordagem pedagógica que utiliza problemas do mundo real, estudos de caso hipotéticos com resultados concretos e convergentes para que os estudantes assimilem o conteúdo planejado e desenvolvam a habilidade de pensar criticamente (MARTINS, 2002).

A PBL como modelo pedagógico, além de facilitar a aprendizagem dos conteúdos previstos, pode proporcionar uma série de outros benefícios. Pode contribuir para que os estudantes aprendam a definir um problema com clareza, a desenvolver hipóteses alternativas, a ceder, a avaliar e utilizar informações de fontes diversas, a alterar hipóteses com base em nova informação, bem como a encontrar soluções que correspondam ao problema e suas respectivas condições.

Segundo Delisle (2000), as raízes da Aprendizagem Baseada em Problemas podem estar ligadas ao movimento progressivo, especialmente à crença de John Dewey de que os professores deviam ensinar apelando aos instintos naturais dos alunos para investigar e criar.

A PBL teve origem na Escola de Medicina da Universidade de McMaster em Hamilton, Ontário, no início da década de 1970 (HERRIED, 2003) e durante muitos anos ficou restrita à formação de profissionais da área médica, sendo uma metodologia desenvolvida para que os alunos, antes mesmo de chegar ao período do internato, pudessem estar em contato com problemas reais. A prática do PBL logo se difundiu pelas faculdades de Medicina de diversos países e depois para outros cursos de graduação e pós-graduação.

Segundo Walsh (2005), a PBL consiste na utilização de um problema para estimular e orientar a aprendizagem dos alunos. A PBL é diferente da “resolução de problemas” (*problem solving*) uma vez que o objetivo da aprendizagem não consiste apenas em resolver o problema que é apresentado. O problema é utilizado para: ajudar os alunos a identificarem as suas próprias necessidades de aprendizagem, enquanto tentam compreender o problema; pensar em conjunto; sintetizar e aplicar informação ao problema; e começar a trabalhar efetivamente para aprender com os membros do grupo e com os tutores.

Porém, a PBL não consiste em simplesmente oferecer um problema aos alunos e ficar aguardando para ver o que eles vão fazer com ele. Os objetivos de aprendizagem devem ser apresentados de forma clara, para assegurar que os alunos atinjam os conhecimentos necessários para a profissão. Desta forma a PBL requer dos tutores organização e dedicação, aperfeiçoamento constante e supervisão criativa para garantir que os alunos estejam no caminho certo e progredindo no seu trabalho(WALSH, 2005).

Woods (1994), da Universidade de McMaster, descreve que a Aprendizagem Baseada em Problemas deve seguir uma seqüência de etapas: “Apresentar o problema” → “Identificar o que precisamos aprender” → “Aprender” → “Aplicar”.

Segundo Sardo (2007), apesar destas etapas serem listadas de uma forma linear, este é um processo dinâmico em um “ir e vir” constante entre as diferentes etapas. Por isso, freqüentemente se assiste a algumas sobreposições das diferentes etapas e por vezes é necessário voltar ao ponto de partida do problema, enquanto o grupo continua o seu trabalho.

Walsh (2005) descreve basicamente a Aprendizagem Baseada em Problemas em sete etapas: a) Identificar o problema; b) Explorar o conhecimento pré-existente; c) Criar hipóteses e mecanismos de atuação possíveis; d) Identificar os conteúdos de aprendizagem; e) Estudar individualmente; f) Re-avaliar e aplicar o novo conhecimento no problema; e g) Avaliar e refletir a aprendizagem. Baseado em Walsh (2005) e Sardo (2007), são apresentadas cada uma destas etapas:

- Identificar o problema: os alunos lêem todo o problema do início ao fim e discutem-no. Eles devem ser incitados a “diagnosticar” o problema de imediato, e necessitam ser encorajados a pensar mais profundamente sobre todos os “porquês, como e quando”.

- Explorar o conhecimento pré-existente: todos os alunos possuem uma base de conhecimentos e várias experiências de vida. Sabe-se que as pessoas retêm mais facilmente o novo conhecimento quando já sabem alguma coisa sobre ele. Assim, esta segunda etapa permite que os estudantes acessem conscientemente às suas próprias compreensões (adquiridas anteriormente) e comecem a aplicá-la em seu próprio beneficio e em benefício dos outros elementos do grupo. Os tutores devem utilizar os seus conhecimentos para assegurar que os estudantes não estão se afastando, ou tomando um caminho não produtivo para a próxima etapa.

- Criar hipóteses e mecanismos de atuação possíveis: o objetivo é manter os estudantes concentrados na compreensão dos conceitos-chave que são ilustrados por cada problema, e isso requer que eles pesquisem/explorem/mergulhem mais profundamente nele. Assim, o tutor deve assegurar que todos os estudantes estejam engajados nesta etapa, e que as hipóteses criadas por eles tenham relação com os objetivos de aprendizagem do problema.

- Identificar os conteúdos de aprendizagem: deverá ficar claro quais serão os seus conteúdos de aprendizagem (individual e grupal). O tutor terá bastante trabalho para ajudar um grupo inexperiente a classificar os conteúdos de aprendizagem, transformá-los em questões centrais, que uma vez mais dizem respeito aos objetivos gerais do problema. Estas questões serão a base da pesquisa dos alunos por fontes/referências e informações, e podem ser evitadas muitas frustrações se estas questões forem claras e precisas.

- Estudar individualmente: o programa educacional deve clarificar se todos os alunos devem se concentrar em todos os conteúdos de aprendizagem, ou se é pertinente que os mesmos selecionem as áreas que irão aprofundar no componente de estudo individual. Deve ser dado algum tempo para o estudo individual antes do próximo tutorial. Assim, é necessário construir um programa educacional que englobe/abranja esta etapa crucial do processo, e não sobrecarregue a agenda dos alunos.

- Re-avaliar e aplicar o novo conhecimento no problema: pesquisas educacionais (assim como o senso comum) sugerem que trabalhar com novas informações, questioná-las, e aplicá-las em diferentes situações, estimulam a sua aplicação no futuro. Assim, os estudantes devem ser incentivados a fazer perguntas, explicar conceitos difíceis, e identificar e compreender os conceitos chaves que podem ser aplicados ao problema. O tutor pode também estimular a leitura dos estudantes fazendo questões que os desafiem a aplicar esses conceitos em contextos ligeiramente diferentes.

- Avaliar e refletir a aprendizagem: antes do problema e do tutorial serem considerados completos é importante que cada estudante e o grupo tenham a oportunidade para refletir sobre o processo de aprendizagem. Isto inclui uma revisão

> da aprendizagem adquirida e representa também uma hipótese para que os elementos do grupo possam dar *feedback* mútuo sobre as contribuições para a aprendizagem, para o processo de grupo e para avaliarem como é que o grupo trabalha em conjunto. Esta etapa ajuda o grupo a realizar pequenos ajustes antes que os problemas cresçam em demasia e não possam ser corrigidos. Resumidamente, as novas aprendizagens são consolidadas para futuras aplicações.

A esta explicação detalhada das diferentes etapas da PBL, Walsh (2005) acrescenta pequenas dicas para os tutores utilizarem, tais como: estimular a análise detalhada do problema, não opinar; usar a experiência para estruturar questões desafiadoras, em detrimento das mini-leituras; concentrar-se na compreensão dos conceitos-chave; ajudar os estudantes a elaborarem detalhadamente as questões que serão pesquisadas; encorajar os estudantes a evitarem pesquisar dentro das áreas que eles já tenham experiência; desafiar os estudantes a aplicarem novos conceitos em diferentes contextos, e reconhecer os conceitos aprendidos previamente quando eles aparecerem novamente; reservar um período para reflexão e *feedback*.

Esta abordagem foi fundamentada nas origens da PBL, nomeadamente nas orientações da Universidade de McMaster e da Universidade de Maastricht, que atualmente são referências a nível mundial. Outra abordagem bastante similar abordada por Oliveira *et al* (2005) é um esquema para PBL para uso em estudo individual, em vez da abordagem tradicional em grupo.

O referido esquema tem como base o modelo pedagógico definido em Labidi e Ferreira (1998a, 1998b *apud* OLIVEIRA *et al*, 2005) e os princípios de projetos organizados por solução de problemas definidos por Barges (2003 *apud* OLIVEIRA *et al*, 2005). A Figura 16 ilustra o esquema definido.

O conjunto de todas as fases do esquema apresentado, desde a fase de Preparação do Aprendiz até a fase de Validação dos Resultados, forma uma sessão de estudo. A fase de Avaliação Final ocorrerá como conseqüência dos resultados apresentados pelo aprendiz nas fases anteriores. Caso ele não tenha apresentado resultados satisfatórios, poderá retornar às fases anteriores, conforme setas indicativas presentes na Figura 16. A resolução de um problema por um aprendiz poderá necessitar de uma única sessão de estudo, ou de várias, o que dependerá do nível do problema a ser trabalhado, bem como do perfil do aprendiz (OLIVEIRA *et al*, 2005).

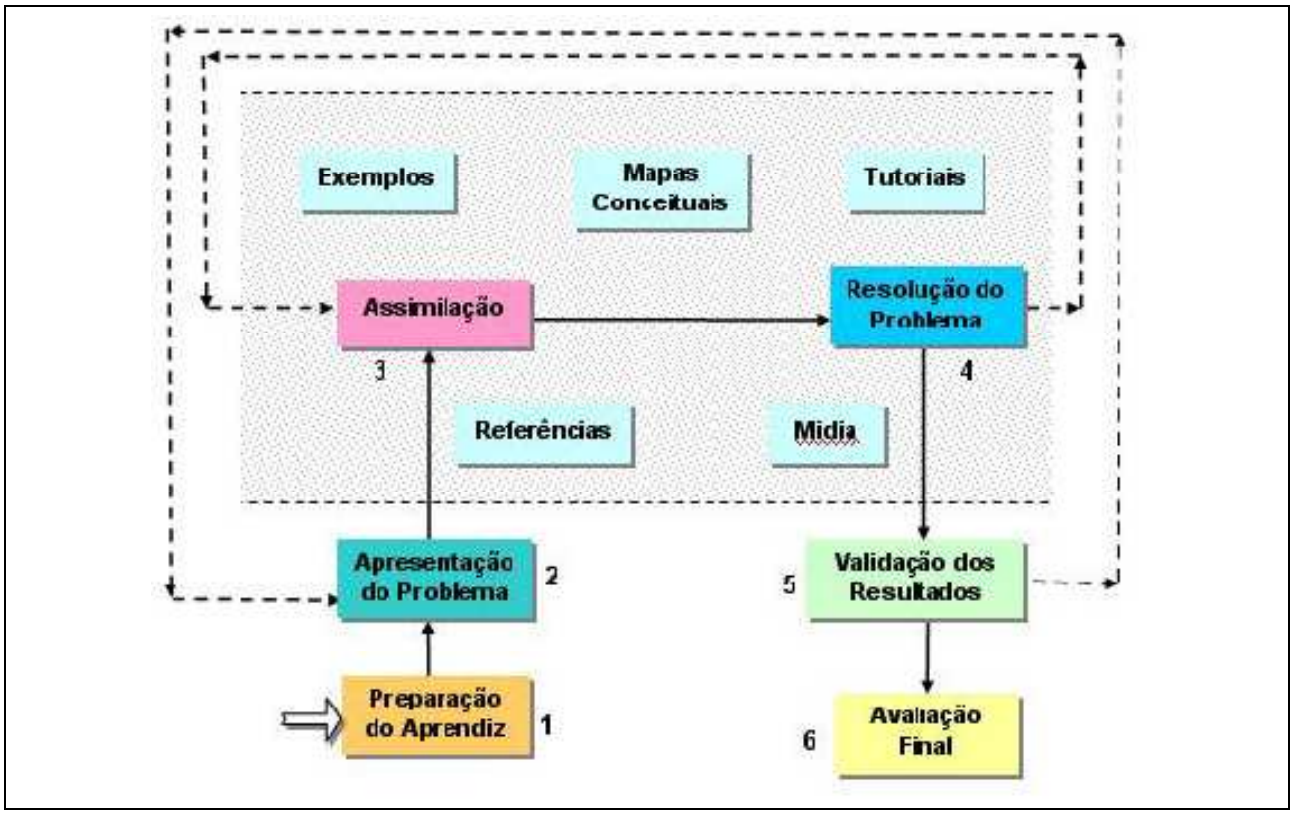


Figura 16. Esquema definido para a PBL.
Fonte: Adaptado de Oliveira *et al* (2005).

Oliveira *et al* (2005) descreve cada uma das etapas:

- Fase de Preparação do Aprendiz: nessa fase, apresenta-se o objetivo da atividade a ser desenvolvida, bem como as etapas da PBL que nortearão o cumprimento do objetivo previamente estabelecido, ou seja, da resolução do problema. Nessa fase, identifica-se ainda o nível inicial do aprendiz, que poderá ser iniciante, intermediário ou avançado, com o propósito de se apresentar um problema de acordo com as características de cada aprendiz.

- Fase de Apresentação do Problema: com base no nível de conhecimento, características individuais e necessidades de cada aprendiz, um problema é apresentado. Cada problema está relacionado com uma unidade pedagógica principal, que engloba os conceitos e procedimentos necessários à sua solução. Dentro do escopo de uma unidade, os problemas são dispostos numa seqüência que vai do mais abrangente, mais complexo, ao mais restrito. A idéia é que os problemas

mais restritos possam solidificar os conceitos estudados para que o aprendiz tenha base para solucionar os problemas mais abrangentes.

- Fase de Assimilação: a fase de assimilação é a fase de análise do problema, geração de hipóteses, identificação de fatos, teoremas etc., relevantes ao contexto, a partir de estudo autodirigido. Nessa fase, o aprendiz faz o levantamento de informações necessárias à solução do problema.

- Fase de Resolução do Problema: com base na fundamentação obtida na fase anterior, o aprendiz tentará resolver o problema. Caso não consiga encontrar uma solução, ele poderá tentar solucionar problemas com menor grau de complexidade. Para isso, o sistema orienta o aprendiz apresentando várias alternativas de percursos dentro das possibilidades de navegação pelo conteúdo oferecido. Adicionalmente, podem ser apresentados conceitos transversais pertinentes ao contexto do problema. Dependendo do seu desempenho, o aprendiz poderá permanecer na fase de resolução do problema, ou retornar à fase de assimilação para reavaliação dos resultados encontrados até o momento. Caso consiga solucionar o problema, o aprendiz passará para a fase de validação dos resultados.

- Fase de Validação dos Resultados: nessa fase, a solução do problema apresentada pelo aprendiz é validada. Caso a solução apresentada esteja incorreta, o aprendiz é orientado a retornar à fase de assimilação ou à fase de apresentação do problema. Por outro lado, se a solução estiver correta, então o aprendiz poderá passar à fase de avaliação final.

- Fase de Avaliação Final: a avaliação final contemplará questões específicas sobre o problema apresentado. Seu objetivo é consolidar o conhecimento adquirido pelo aprendiz durante o processo de resolução dos problemas propostos. A idéia é que esta fase ocorra somente quando o sistema obtiver parâmetros das fases anteriores que demonstrem que o aprendiz foi bem sucedido em seu objetivo.

Observa-se que as duas abordagens apresentadas são bastante similares, porém a primeira (WALSH, 2005) tem foco na área de medicina para trabalhos em grupo, enquanto a segunda abordagem (OLIVEIRA *et al*, 2005) tem foco em ambiente virtuais de ensino para trabalhos individuais.

Para Oliveira *et al* (2005), a construção de ambientes computacionais baseados na Aprendizagem Baseada em Problemas enfrenta, dentre outros, dois grandes desafios. O primeiro refere-se à falta de um padrão prático que permita o reconhecimento de uma atividade de aprendizagem como PBL. O segundo, comum a diversos outros tipos de ambientes de aprendizagem, refere-se a como modelar e representar a interação do aprendiz com o ambiente.

O método PBL evidencia como proposta o estímulo do pensamento crítico, habilidades para solução de problemas e a aprendizagem de conceitos na área em questão. E se diferencia das abordagens convencionais, pois o ensino é centrado no estudante, iniciando-se com o uso de problemas para direcionar, motivar e focar a aprendizagem. O problema é uma situação real ou uma simulação próxima da realidade, abrangendo várias áreas de conhecimento.

Diante do exposto, neste trabalho pretende-se aplicar a PBL como abordagem para o ensino da técnica de RBC, seguindo as etapas propostas por Oliveira *et al* (2005), que é focada em estudo individualizado, onde o aluno deve ser capaz de realizar exercícios propostos utilizando como base as funcionalidades da ferramenta.

3 DESENVOLVIMENTO

Este trabalho tem como objetivo principal elaborar uma ferramenta para ensino e desenvolvimento de sistemas de Raciocínio Baseado em Casos, visando auxiliar os usuários e desenvolvedores com ou sem experiência, na estruturação da representação do conhecimento, na definição das métricas de similaridade, na busca dos casos similares e na criação de regras para retenção de novas informações na base de casos.

3.1 ANÁLISE E PROJETO

A ferramenta visa disponibilizar um ambiente onde os usuários possam entender o funcionamento da técnica de RBC, através dos tutoriais e de um personagem que está disponível prestando dicas sobre como realizar as definições, e também elaborar novos sistemas de RBC interagindo com o ambiente. A interface com o usuário segue padrões de ergonomia e usabilidade de forma intuitiva, informando o usuário de todas as possibilidades que o ambiente dispõe e os próximos passos que devem ser realizados.

O material de auxílio ao usuário foi desenvolvido visando ajudar na definição de qual forma de representação do conhecimento pode ser mais adequada para o problema que se pretende desenvolver, qual métrica de similaridade pode ser mais apropriada para cada tipo de informação no sistema, como serão armazenadas as informações na base de casos e também como poderão ser adaptados os novos casos para serem adicionados a base de casos.

Com as definições foi possível elaborar o esquema apresentado na Figura 17, que ilustra os três principais componentes que devem ser definidos na elaboração de um sistema RBC: a) Representação do Conhecimento e Base de Casos; b) Estrutura e Recuperação das informações; e c) Reutilização e Adaptação dos Casos.

Assim o usuário deve desenvolver o sistema de RBC e gerar a base de casos, seja como Banco de Dados Relacional ou Arquivos Texto, permitindo definir a forma como os dados serão representados e estruturados, como também os atributos que irão compor cada caso.

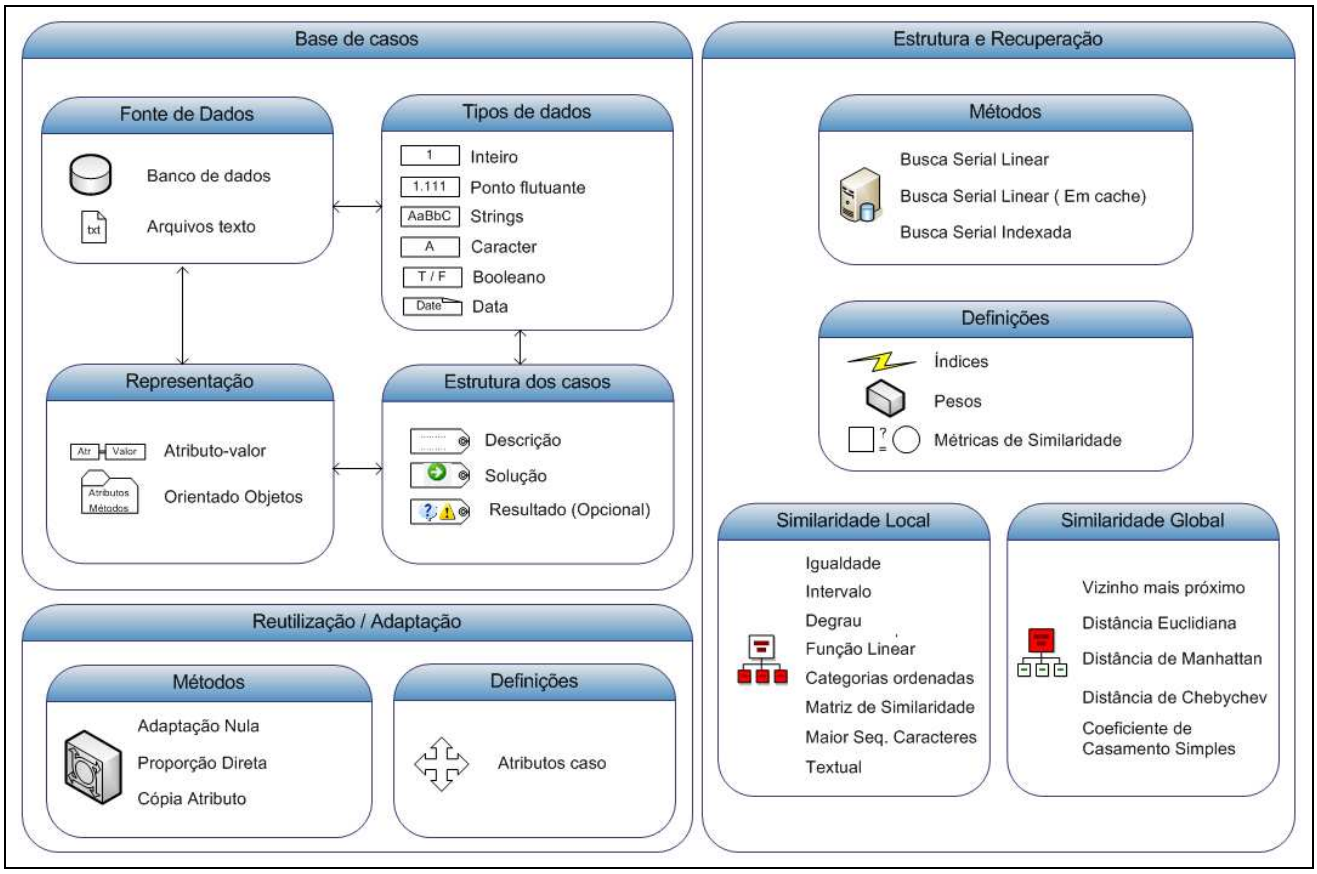


Figura 17. Esquema da estrutura do ambiente.

Para recuperar as informações do sistema RBC, o usuário pode definir os métodos de recuperação e quais atributos serão os índices (discriminantes) que diferenciarão cada caso dos demais e seus respectivos pesos para serem utilizados no cálculo da similaridade.

Para o cálculo da similaridade serão disponibilizadas diferentes métricas locais e globais, sendo que novas métricas podem ser adicionadas à ferramenta. Como delimitação do escopo do trabalho foram definidas as métricas apresentadas na Figura 17, que formam o conjunto das principais métricas utilizadas em sistemas de RBC.

Para adaptação dos casos o usuário pode definir os métodos que serão aplicados nos atributos do caso ou optar pela Adaptação Nula, que não realiza nenhuma alteração nos casos recuperados.

Partindo destas definições foi possível elencar as principais funcionalidades disponibilizadas na ferramenta, conforme apresentado na Figura 18.

Figura 18. Funcionalidades do Ambiente

Na ferramenta o usuário pode recuperar as estruturas de representação do conhecimento já utilizadas em outras aplicações, como tabelas de bancos de dados relacional. Todas as configurações ou definições do sistema RBC podem ser realizadas através da interface gráfica, possibilitando consultar o tutorial, as ajudas contextualizadas e as mensagens apresentadas pelo personagem.

Os testes dos sistemas de RBC desenvolvidos são realizados dentro da própria ferramenta, onde se pode visualizar o ciclo RBC e inspecionar a execução de cada etapa do ciclo. Também é possível exportar o sistema desenvolvido, permitindo a execução em outros ambientes e possibilitando a geração de uma interface personalizada.

Ainda é disponibilizada aos tutores (professores) uma área para definição de problemas (exercícios) para serem desenvolvidos (resolvidos) pelos alunos utilizando a ferramenta. Para o aluno desenvolver estes exercícios será utilizada a abordagem de ensino Aprendizagem Baseada em Problemas (PBL), sendo sua aplicação apresentada na próxima seção.

3.2 APLICAÇÃO DA PBL

Como um dos objetivos deste trabalho é o ensino da técnica de RBC, foi incluída uma área específica para alunos desenvolverem exercícios, visando explorar e aplicar os conhecimentos obtidos na academia. Desta forma foi definida como abordagem pedagógica a Aprendizagem Baseada em Problemas ou *Problem Based Learning* (PBL), sendo utilizada como base o trabalho de Oliveira *et al* (2005), que trata a PBL como atividade individualizada para cada aluno.

A PBL é aplicada na realização dos exercícios, onde os alunos devem passar por cada uma das fases propostas no trabalho de Oliveira *et al* (2005): a) Preparação do Aprendiz; b) Apresentação do Problema; c) Assimilação; d) Resolução do Problema; e) Validação dos Resultados; e f) Avaliação Final.

Na etapa de preparação do aprendiz a ferramenta disponibiliza um tutorial com informações detalhadas sobre o funcionamento da técnica de Raciocínio Baseado em Casos, onde o usuário pode obter um embasamento teórico mínimo sobre o tema. Caso o usuário já tenha conhecimento sobre sistemas de RBC e não deseje consultar o tutorial, pode-se passar para a próxima etapa da PBL, que é a apresentação do problema.

Na etapa de apresentação do problema é exibida a descrição do exercício (Problema) criado pelo tutor para que o aluno possa visualizar as informações sobre o problema a ser resolvido, levando-o a entender o objetivo que deve ser alcançado. Na elaboração das atividades o tutor pode relacionar dicas do problema, para facilitar o entendimento pelo aluno, sendo que estas podem ser acessadas por um botão específico e são exibidas na tela seqüencialmente.

Após a apresentação do problema, o aluno passa para etapa de assimilação, onde é exibida uma tela para descrever, de forma textual, quais atributos do sistema RBC ele conseguiu identificar como sendo relevantes (descrição do problema e a respectiva solução). Caso o aluno tenha dificuldades em definir os atributos, poderá voltar à etapa anterior para consultar a descrição do problema e obter as dicas novamente, como também poderá consultar o tutorial.

A partir dos atributos o usuário poderá seguir para etapa de resolução do problema, onde deve definir as métricas de similaridade para cada atributo definido na etapa anterior, bem como as adaptações que serão necessárias, caso existam. Esta etapa o usuário utiliza a própria tela de similaridade da ferramenta (a mesma tela utilizada fora da realização do exercício), possibilitando ainda definir o método de recuperação dos casos.

Com a resolução do problema, o usuário realiza a validação dos resultados, onde visualiza o funcionamento do sistema desenvolvido, através da funcionalidade de execução que apresenta cada etapa do ciclo RBC, como também visualiza as informações da base de casos, caso possua alguma informação já cadastrada.

Como forma de avaliação, o usuário pode gerar um relatório com as definições realizadas durante a execução das atividades, e estas informações são apresentadas ao tutor que realizará a

avaliação final. Neste relatório estão definições tais como atributos identificados, métricas de similaridade utilizadas, adaptações e o número de ciclos RBC executados.

Assim a ferramenta pode ser utilizada por professores da disciplina de Inteligência Artificial para ensinar a técnica de Raciocínio Baseado em Casos. Exemplos de exercícios são disponibilizados e podem ser utilizados para exercitar os conceitos de RBC, bem como servirem como base para o desenvolvimento e disponibilização de novos exercícios para os alunos e a comunidade científica.

Um exemplo de exercício, extraído de Wangenheim e Wangenheim (2003) é apresentado a seguir:

> "Assuma que a empresa de viagens WORLD TRAVEL solicitou a você, desenvolvedor de aplicações RBC, o desenvolvimento de um sistema que auxilie de forma inteligente clientes a encontrar pacotes de viagens no site da empresa na Internet. Experiências anteriores da empresa WORLD TRAVEL, ao tentar disponibilizar seu catálogo de pacotes de viagem online utilizando enfoques de banco de dados tradicionais, no entanto, falharam. Muitos clientes queixaram-se de, que em muitos casos, o sistema não foi capaz de encontrar nenhum pacote de viagens, ou encontrou uma lista excessivamente grande. Agora, a empresa está buscando um enfoque inteligente que mesmo quando não existir um pacote de viagem que satisfaça exatamente todos os desejos do cliente, ofereça um pacote similar de forma a tentar satisfazer o cliente com opções. Em função disso, a tecnologia RBC foi escolhida para ser adotada no desenvolvimento desse site inteligente."

Seguindo este exemplo, os alunos devem identificar em website de empresas de viagens os atributos (variáveis) que estão envolvidos no problema, como origem, destino, acomodações, etc, com o objetivo de criarem um sistema de busca de pacotes de viagem inteligente. Com o exercício concluído o usuário terá exercitado a técnica e terá desenvolvido o sistema proposto, conseguindo colocar o conhecimento e aprendizagem em um contexto do mundo real.

Neste contexto o professor tem o papel de supervisor do processo, acompanhando os alunos durante o desenvolvimento do trabalho sem a necessidade de intervenção contínua, porém interagindo quando necessário, atuando de forma direta no fim do processo para avaliar o sistema desenvolvido pelos alunos.

3.3 MODELAGEM DA FERRAMENTA

Antes do início da modelagem foi feita a definição das tecnologias que seriam utilizadas. Para o uso do *framework* jColibri versão 2.1 utiliza-se a linguagem de programação Java, sendo desta forma necessário o Kit de Desenvolvimento Java (SDK) versão 1.6.0.

A partir da definição destas tecnologias elaborou-se um Diagrama de Casos de Uso. Este diagrama descreve as funcionalidades propostas na ferramenta, representado por unidades discretas da interação entre um usuário e o sistema. Os casos de uso são tipicamente relacionados a "atores", que podem ser um humano ou entidade máquina que interage com o sistema para executar um determinado trabalho. Na Figura 19 é apresentado o Diagrama de Caso de Uso.

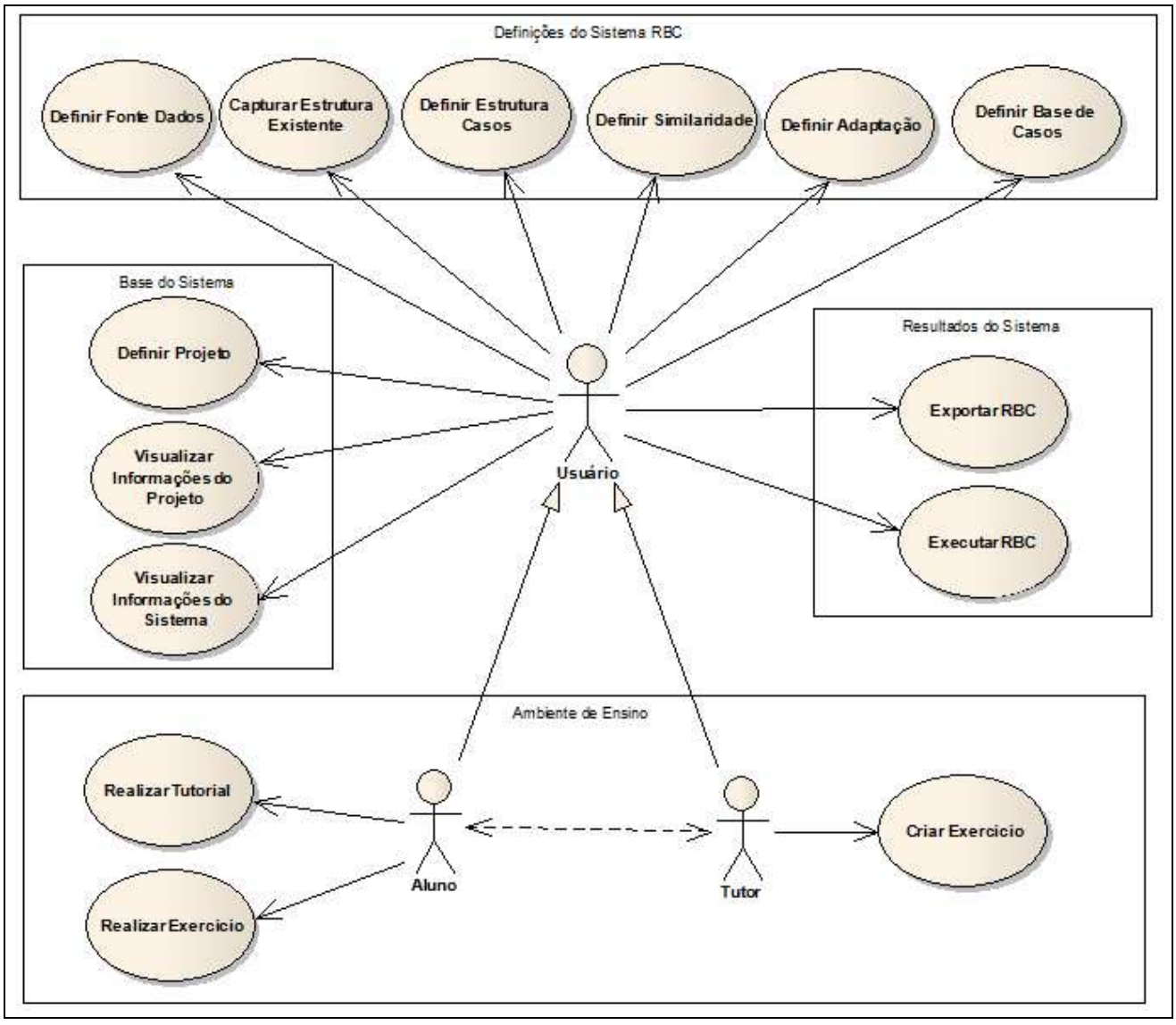


Figura 19. Diagrama de Caso de Uso

O sistema é composto por três atores: Usuário (possibilita a execução de todas as funcionalidades do ambiente), Aluno (possibilita a visualização do tutorial e a realização dos exercícios propostos) e Tutor (responsável por criar e disponibilizar novos exercícios para a ferramenta e supervisionar/interagir com o aluno).

Na ferramenta foram levantados quatro grupos de casos de uso: base do sistema (responsáveis pela manipulação das informações dos projetos criados), definição do sistema RBC (responsáveis pelas definições do sistema RBC), resultados do sistema (responsáveis pela execução das aplicações desenvolvidas e pela exportação do projeto) e ambiente de ensino (responsáveis pela criação e realização dos exercícios, como também a navegação pelos tutoriais).

A seguir são descritos os casos de uso que compõe o grupo base do sistema:

- Definir Projeto: o usuário pode criar um novo projeto, ou abrir um projeto existente. O usuário também terá disponível as opções “salvar” e “salvar como” para armazenar as definições realizadas e também fechar o projeto atual.

- Visualizar Informações do Projeto: o usuário pode visualizar as informações do projeto, através da opção de visualizar informações, onde estão disponíveis informações sobre definições já realizadas para o projeto atual. O usuário também poder imprimir estas informações, em forma de relatório, onde são exibidas as definições realizadas do sistema RBC.

- Visualizar Informações do Sistema: o usuário pode visualizar informações como versão atual do sistema, bem como os dados para contato dos desenvolvedores e qual foi o objetivo em desenvolver a ferramenta.

O grupo definição do sistema RBC é composto pelos seguintes casos de uso:

- Definir fonte de dados: nesta opção o usuário pode definir qual a estrutura utilizada para armazenar as informações da base de casos, ou seja, onde será realizada a persistência dos dados. Existe a possibilidade de definir bases de casos no formato de arquivos textos e banco de dados relacional.

- Capturar Estrutura Existente: caso o usuário utilize uma base de casos que já está definida (já utilizada em outras sessões). No caso de bancos de dados, pode ser recuperada a estrutura das informações, onde a ferramenta é capaz de identificar estruturas como tabelas, campos e relacionamentos existentes, devendo o usuário definir quais os campos serão utilizados no sistema RBC.

- Definir Estrutura dos Casos: o usuário pode definir a estrutura que compõe o caso, definindo através de atributos a descrição do problema, a descrição da solução e o resultado. Com a conclusão das definições é possível criar estas estruturas na fonte de dados informada pelo usuário, ou caso seja definida uma estrutura existente, esta será automaticamente relacionada com as definições realizadas.

- Definir Similaridade: para definição da similaridade o usuário tem duas opções. A primeira é definir a similaridade local, que determina como é realizada a comparação (cálculo da similaridade) entre cada atributo do caso, realizado através de uma função que retorna um valor numérico e também se pode definir os pesos de cada atributo (utilizado para destacar atributos mais significantes). A segunda opção é a definição da similaridade global, que também é gerada através de uma função, além do usuário definir qual o parâmetro para exibição dos casos (quantidade máxima de casos ou similaridade mínima).

- Definir Adaptação: o usuário pode definir em quais campos haverá adaptação das informações ao realizar o ciclo RBC, definindo os atributos e qual a forma de adaptação será utilizada. O usuário também pode definir que não haverá adaptação no caso recuperado.

- Definir Base de Casos: o usuário tem a opção de definir o método de recuperação dos casos, podendo selecionar entre busca linear, indexada e linear em cachê. O usuário também pode definir filtros iniciais, onde o sistema realiza a seleção inicial dos casos que satisfazem este filtro e realiza o cálculo da similaridade entre os casos resultantes.

No grupo Resultados do Sistema estão os casos de uso:

- Executar RBC: após a definição da estrutura do sistema RBC o usuário realiza testes com o sistema RBC desenvolvido, validando suas definições e verificando o desempenho. Durante a execução o usuário pode acompanhar cada etapa do ciclo RBC, desde a recuperação até a retenção de novos casos na base. O usuário também poderá visualizar as informações da base de casos, possibilitando a inclusão de novos casos diretamente na base.

- Exportar RBC: o usuário após executar os testes no sistema RBC pode exportar as definições realizadas no ambiente, possibilitando desta forma a integração com aplicações externas. São disponibilizados os arquivos de configuração e a biblioteca (em conjunto com a biblioteca do *framework* jColibri).

O grupo Ambiente de Ensino é composto pelos seguintes casos de uso:

- Realizar Tutorial: o usuário aluno poderá consultar os tutoriais disponibilizados no ambiente para absorver os conteúdos relacionados a sistemas RBC, possibilitando o desenvolvimento de sistemas dentro e fora do ambiente.

- Realizar Exercício: o usuário aluno pode realizar os exercícios propostos para aperfeiçoar os conhecimentos sobre RBC. Como resultado da realização do exercício o aluno terá o sistema RBC desenvolvido e realizando testes e verificações.

- Criar Exercício: o usuário tutor pode criar novos exercícios para serem desenvolvidos pelos alunos com a ferramenta. O tutor deve descrever o cenário que identifica a situação onde o RBC será aplicado e dicas para cada etapa de realização do exercício, que podem ser solicitados pelo aluno.

Conforme definição dos casos de uso, para utilização da ferramenta é definido um projeto para cada sistema que se deseja desenvolver. O usuário deve obrigatoriamente utilizar um projeto de cada vez, e todas as definições realizadas serão armazenadas em formato XML, utilizando para isso a API chamada XStream que realiza a serialização de objetos Java em XML e o processo inverso, carregando de um arquivo XML os objetos Java.

Para exemplificar como será o funcionamento foi criado um Diagrama de Atividades onde é possível visualizar a ordem das etapas que devem ser realizadas, onde as circunferências indicam o início e término da atividade, as setas indicam o fluxo das atividades e as caixas retangulares indicam as atividades (internamente as opções disponíveis em cada atividade), que correspondem aos casos de uso citadas anteriormente. Na Figura 20 é apresentado o Diagrama de Atividade correspondente ao desenvolvimento de uma aplicação, caso o usuário opte por desenvolver sem estar relacionado com um exercício proposto.

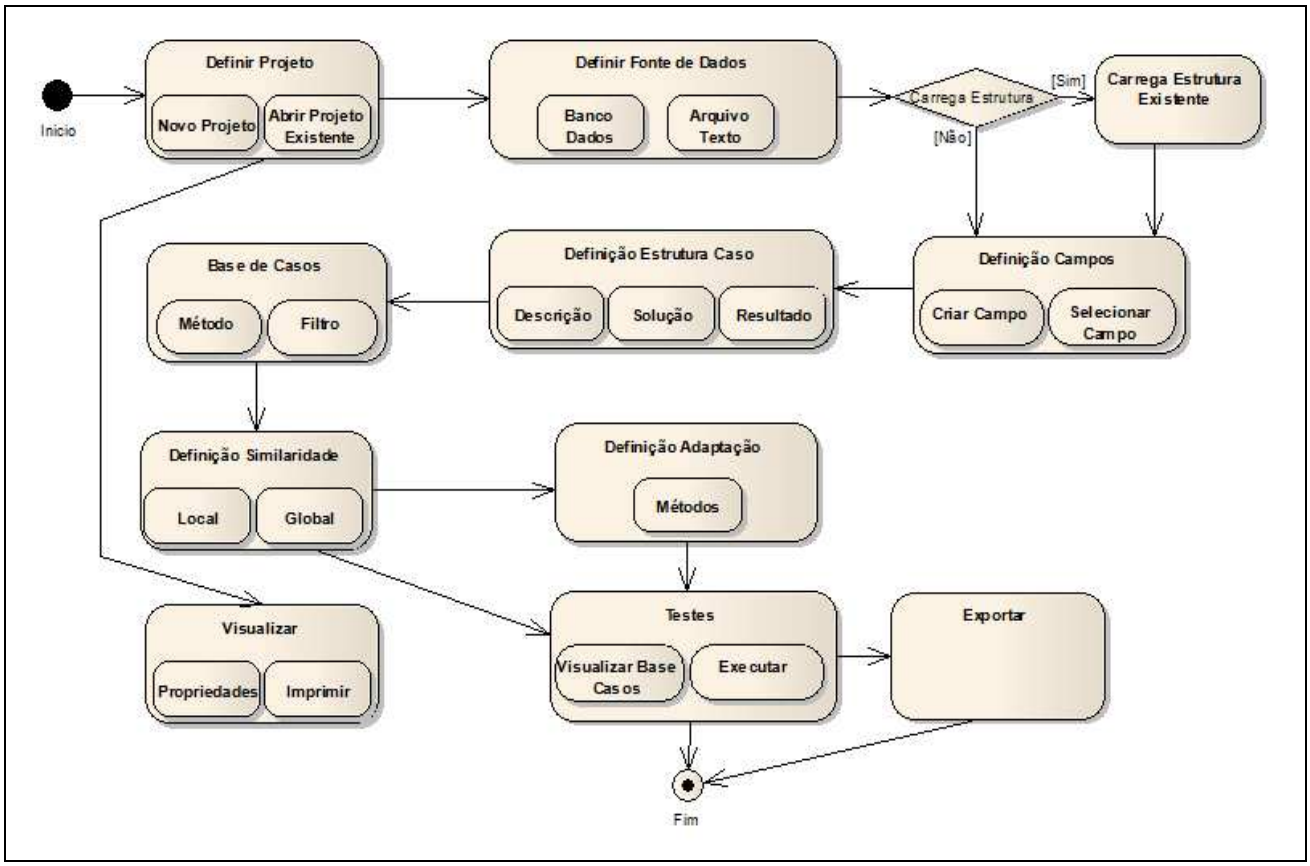


Figura 20. Diagrama de Atividade para desenvolvimento de uma aplicação

Na Figura 21 é apresentado o Diagrama de Atividades correspondente à criação de novos exercícios. Esta funcionalidade foi desenvolvida para tutores, porém está disponível para qualquer usuário utilizar.

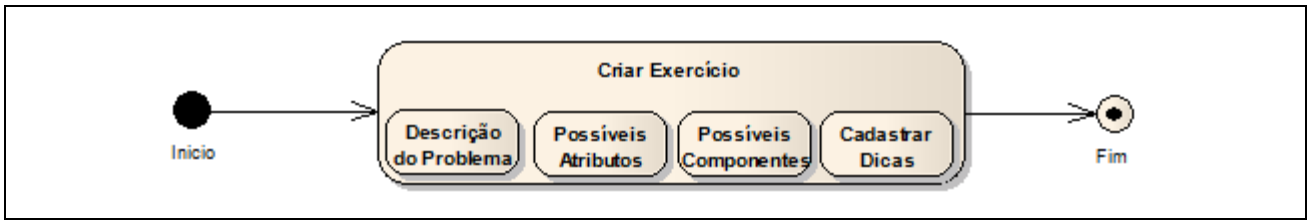


Figura 21. Diagrama de Atividade para criação de exercícios

Na Figura 22 é apresentado o Diagrama de Classes correspondente a realização dos exercícios dentro do ambiente. Inicialmente o Aluno pode optar por realizar o tutorial para obter mais informações sobre RBC, ou caso já tenha o conhecimento necessário, pode iniciar diretamente os exercícios. Após o início dos exercícios o Aluno será direcionado pelas atividades baseado na abordagem PBL, onde passa por cada uma das etapas.

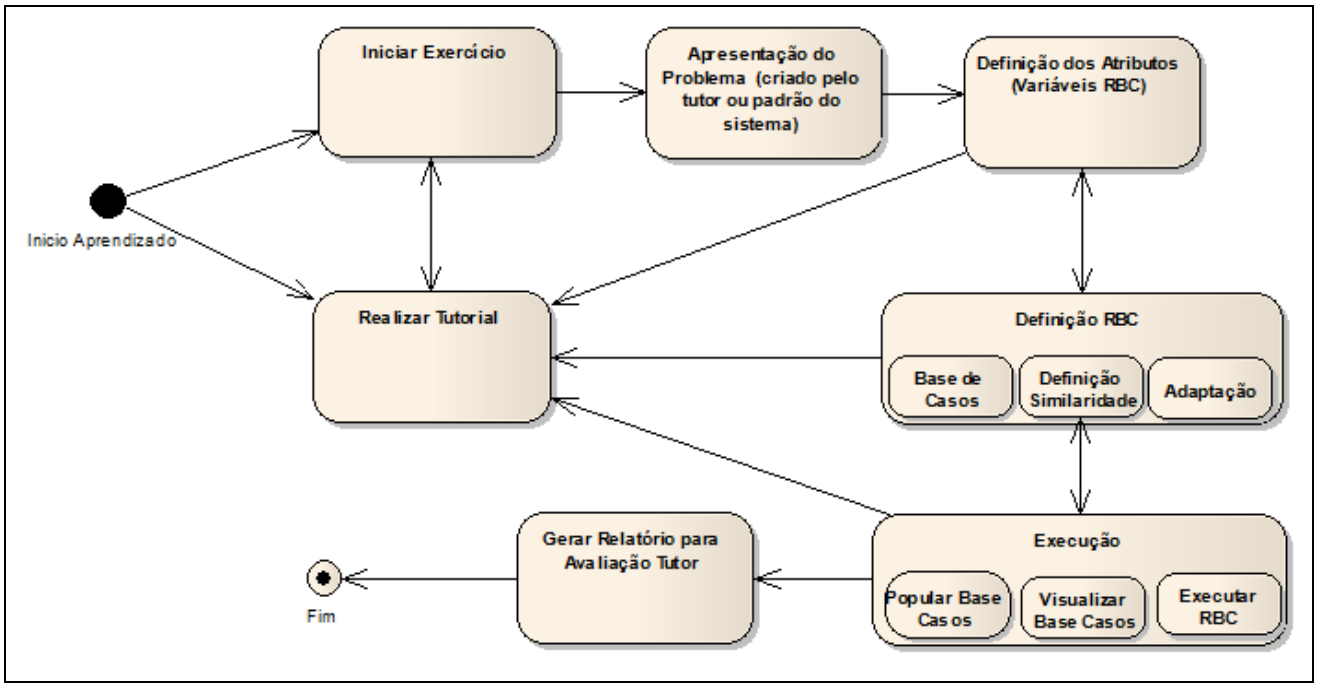


Figura 22. Diagrama de Atividade para realização de exercício

Após ser apresentada a descrição do problema, o aluno realiza as definições do sistema RBC, e caso seja necessário, pode retornar às etapas anteriores através das funcionalidades disponíveis na tela. Com as definições o usuário pode testar e avaliar o sistema. Como conclusão do exercício o aluno deve imprimir o relatório que pode ser entregue ao tutor para uma possível avaliação do sistema desenvolvido.

Para modelar e documentar a estrutura da ferramenta foram utilizados dois diagramas da linguagem UML, o Diagrama de Pacotes e o Diagrama de Classes. O Diagrama de Pacotes descreve os pacotes do código fonte divididos em agrupamentos lógicos apresentando as dependências entre estes, ou seja, pacotes podem depender de outros pacotes. Este diagrama é muito utilizado para ilustrar a arquitetura de um sistema mostrando o agrupamento de suas classes.

Este diagrama foi utilizado para apresentar a estrutura completa das classes que compõe a ferramenta, pois a apresentação de todas as classes em um único diagrama torna-se inviável. Desta forma na Figura 23 são apresentados todos os pacotes e classes..

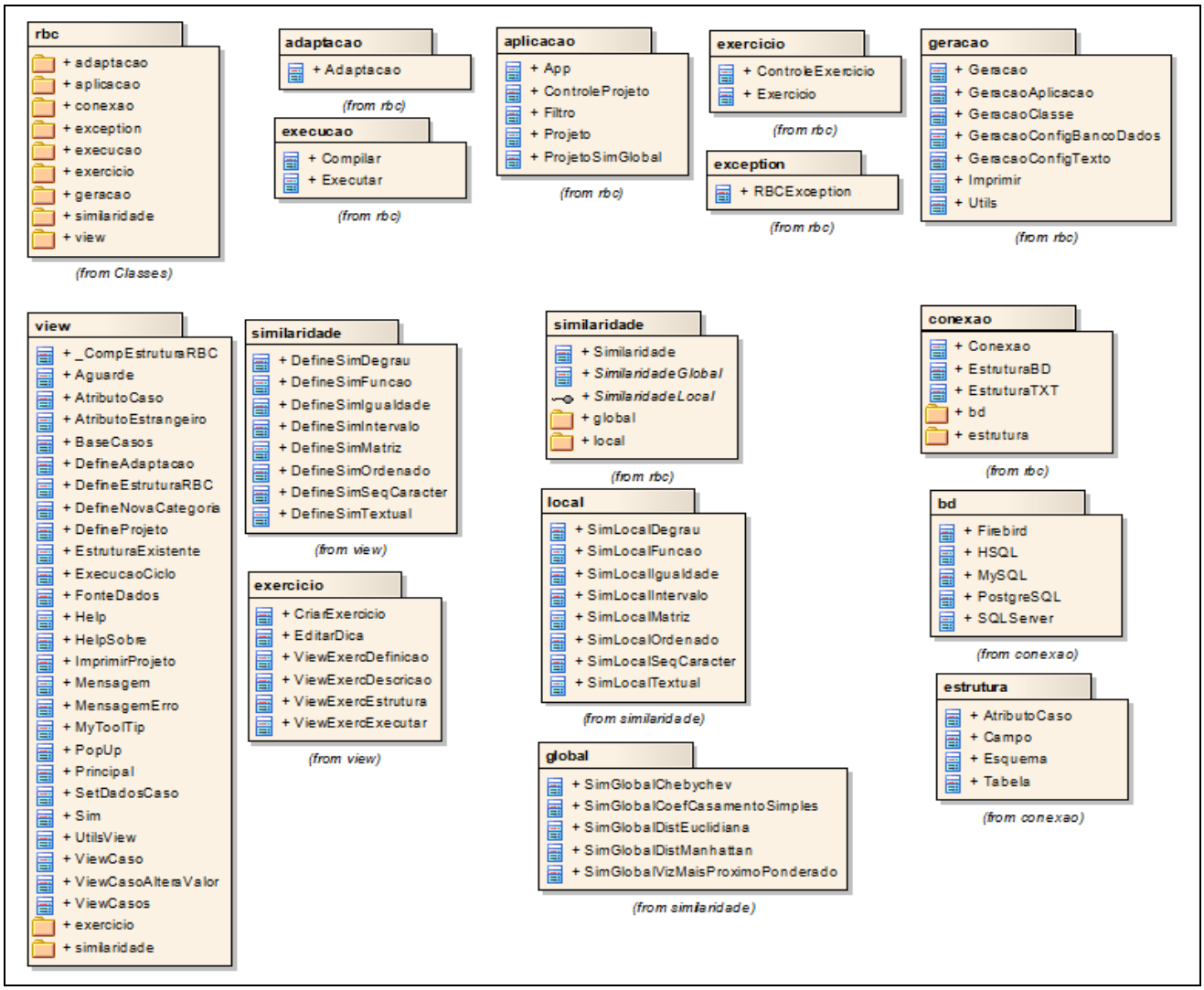


Figura 23. Diagrama de Pacotes.

Para apresentar em detalhes a estrutura da ferramenta, o Diagrama de Pacotes foi desmembrado em três digramas de classe. O Diagrama de Classes é uma representação da estrutura e relações das classes que servem de modelo para objetos. Neste diagrama são definidas as classes que o sistema necessita para seu funcionamento.

Na Figura 24 são apresentas as classes de Controle da aplicação, onde são apresentadas as principais classes utilizadas para o gerenciamento da ferramenta. A principal classe é **App**, que inicia a ferramenta invocando a Tela Principal. A classe que merece destaque é **Projeto** que representa as informações básicas do projeto (armazena as definições realizadas na ferramenta). É esta classe que é armazenada no formato XML para persistir as configurações do projeto.

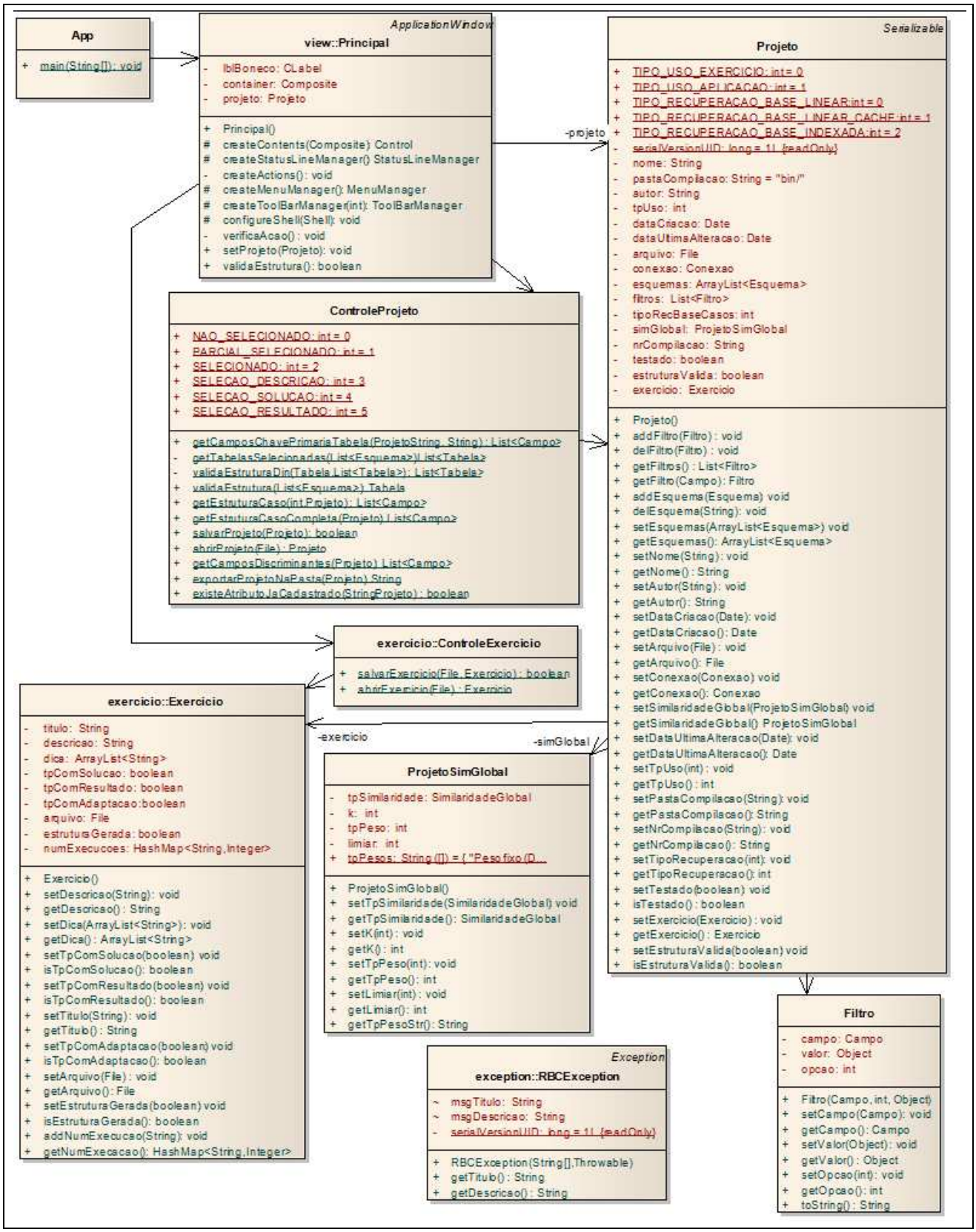


Figura 24. Diagrama de Classes de Controle

Na Figura 25 são apresentadas as classes responsáveis por manter as informações de Conexão com as fontes de dados (Banco de Dados e Arquivos TXT), como também manter as estruturas que armazenam as informações.

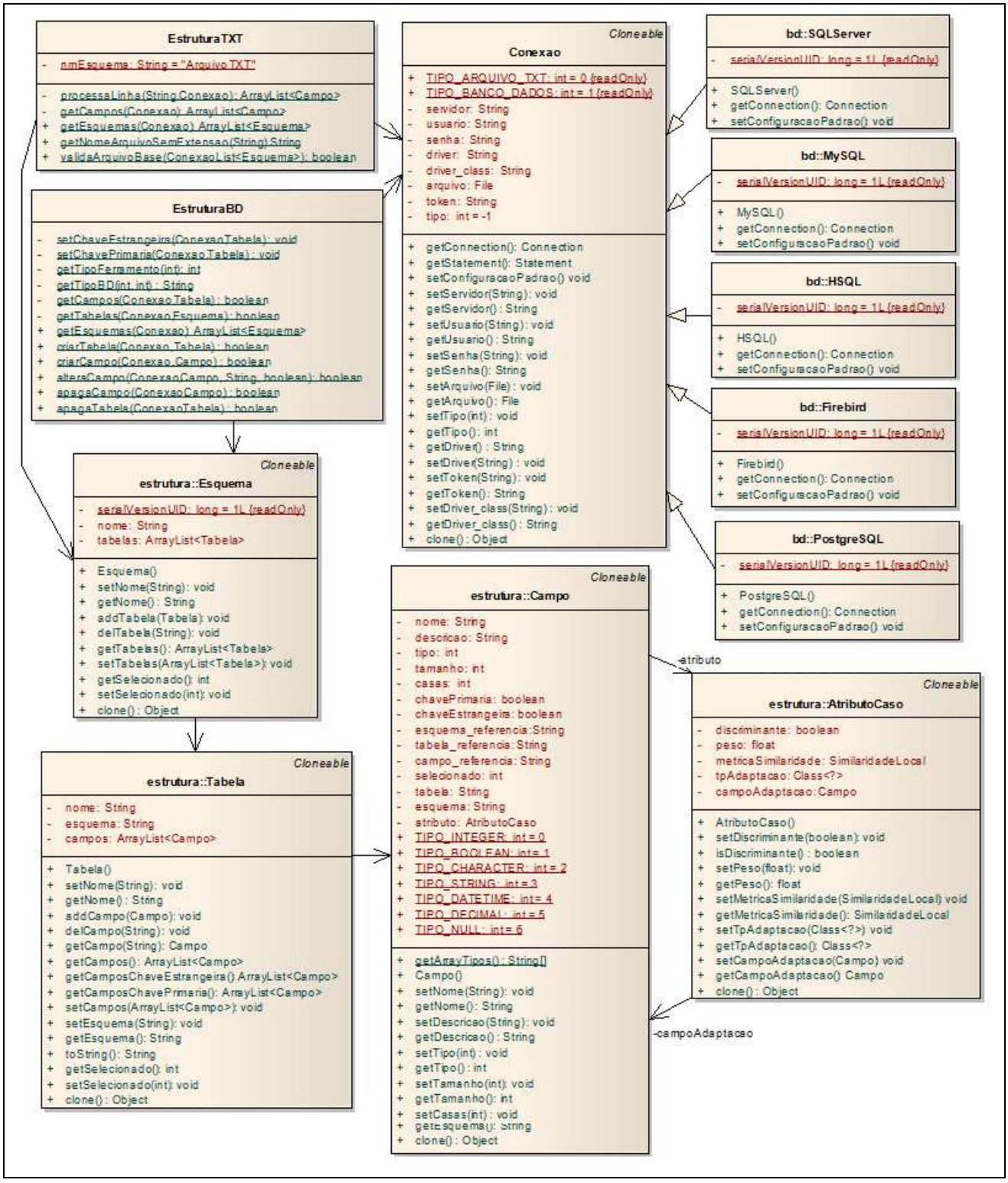


Figura 25. Diagrama de Classes da Estrutura

Na ferramenta foram disponibilizados por padrão as configurações e *drivers* de conexão com os principais bancos de dados existentes, sendo para isso desenvolvida uma Classe específica para cada tipo. As classes **Esquema**, **Tabela** e **Campo** tem as estruturas necessárias para armazenar os metadados [17]de um Banco de Dados relacional, sendo a principal fonte de dados que pode ser utilizada. Quando utilizada a opção Arquivo TXT, são armazenas as estruturas de forma similar às utilizadas em um Banco de Dados. A Classe **AtributoCaso** é utilizada para armazenar as definições quando um campo (atributo) é definido como discriminante no sistema de RBC.

Na Figura 26 são apresentadas as classes correspondentes as métricas de similaridade implementadas. Na parte superior da figura está a Classe **SimilaridadeGlobal**, que deve ser estendida (*extends*) para implementação de novas métricas de similaridade global.

A Classe **SimilaridadeLocal** define uma interface que deve ser implementada pelas métricas de similaridade local, onde cada métrica possui seus parâmetros específicos que podem ser definidos.

A classe **Similaridade** é responsável por gerenciar as métricas de similaridade. Para adicionar uma nova métrica (local ou global) à ferramenta, o usuário deve desenvolver a classe e adicioná-la na respectiva variável desta classe.

Além das classes apresentadas nos diagramas, existe a classe **Adaptacao**, que gerencia os métodos de adaptação. Também não foram detalhadas as classes do pacote **geracao**, que são responsáveis por gerar as classes e os arquivos de configuração que são utilizados pelo *framework* jColibri, as classes do pacote **execucao**, que são responsáveis por compilar os arquivos gerados e executá-los e as classes do pacote **view** que correspondem as telas do sistema.

[17] Metadados (DD ou Dicionário de dados), são dados sobre outros dados. Um item de um metadado pode dizer do que se trata aquele dado, geralmente uma informação inteligível por um computador. Os metadados facilitam o entendimento dos relacionamentos e a utilidade das informações dos dados.

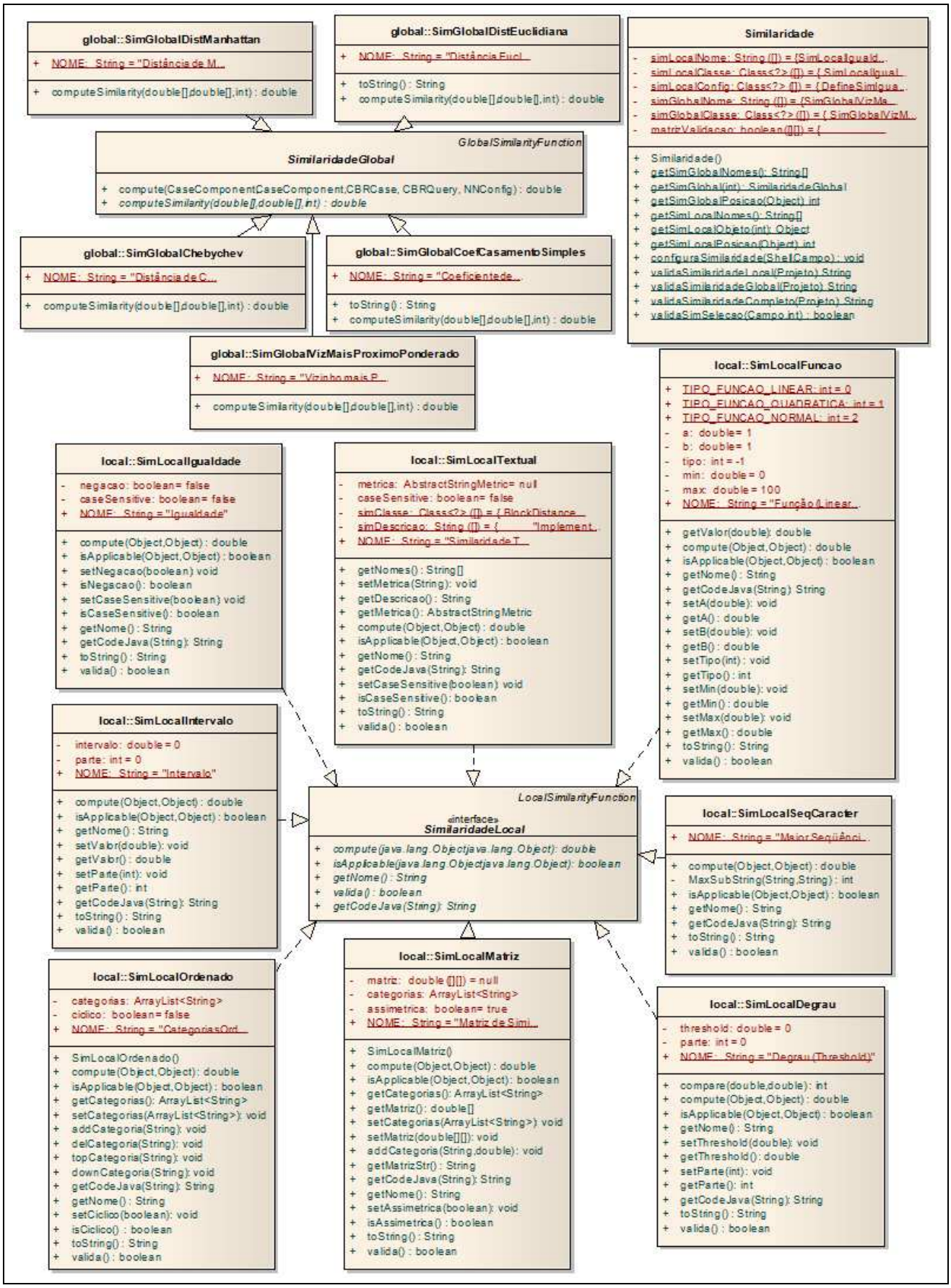


Figura 26. Diagrama de Classes das Métricas de Similaridade

A ferramenta gera em tempo de execução todas as classes do projeto, gravando numa pasta temporária da aplicação. Estas classes são criadas no padrão do *framework* jColibri e são compiladas também em tempo de execução, para então serem executadas. Além das classes também são gerados os arquivos de configuração XML, que armazenam informações de conexão com Base de Dados e relacionamentos entre as classes.

Ao exportar a aplicação estas classes e arquivos de configuração são copiados para pasta do projeto, onde também é disponibilizada uma classe principal que pode ser executado o sistema de RBC em modo console. Para esta execução é utilizada uma biblioteca onde estão os arquivos necessários para execução do sistema sem o uso da ferramenta.

A partir da modelagem apresentada foi possível iniciar o desenvolvimento, onde a descrição das funcionalidades é apresentada na seqüência.

3.4 IMPLEMENTAÇÃO

Seguindo os protótipos de tela elaborados durante a concepção do projeto, foi realizado o desenvolvimento utilizando o Ambiente de Desenvolvimento Integrado (IDE) Eclipse. No início do desenvolvimento foi realizada uma análise das APIs (*Application Programming Interfaces*) gráficas para desenvolvimento com a linguagem de programação Java. Foram analisadas as APIs AWT (*Abstract Window Toolkit*), Swing e SWT (*Standard Widget Toolkit*). A API SWT se mostrou mais apropriada para o desenvolvimento, pois possui uma gama variada de componentes que auxiliam na elaboração de interfaces gráficas.

SWT originalmente foi desenvolvido pela IBM e hoje é mantido pela Fundação Eclipse[18]. A diferença em relação as outras APIs está na composição dos elementos gráficos que acessam diretamente as bibliotecas gráficas do sistema operacional, o que torna a interface com bom desempenho e mantém a compatibilidade com os diversos sistemas operacionais que suportam a linguagem de programação Java.

Na Figura 27 é apresentada a tela principal, exibida no momento que é realizado o acesso.

---

[18] http://*www.**eclipse**.org*

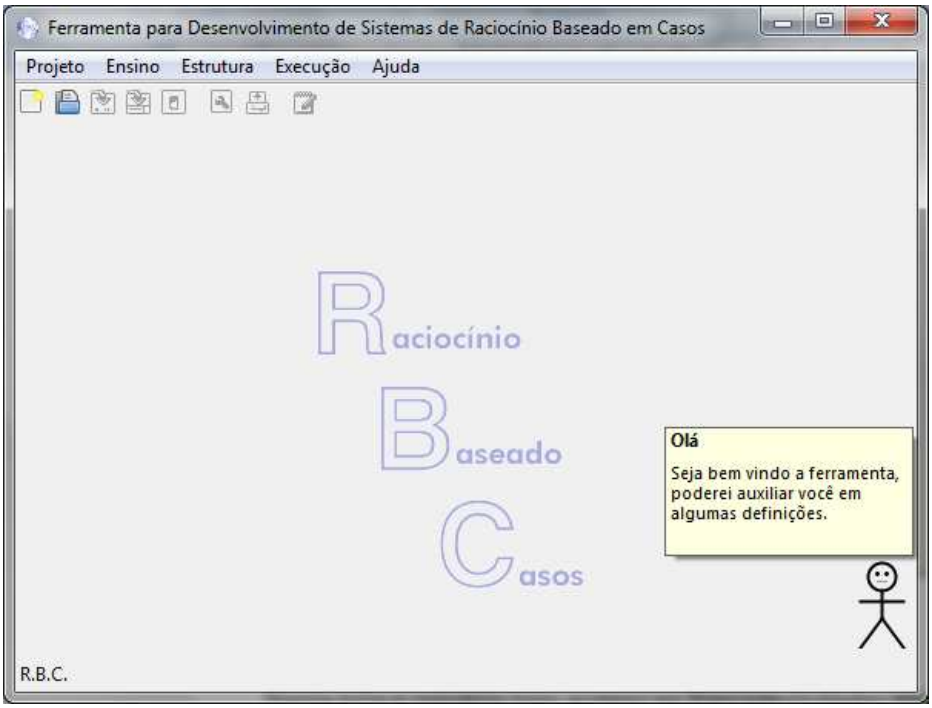


Figura 27. Tela principal da ferramenta

Nesta tela (Figura 27) o usuário tem acesso as funcionalidades através do menu superior. Também existe uma barra de ferramentas na parte superior da tela onde se tem acesso às funcionalidades de manipulação dos projetos. Na parte inferior direita é exibido o personagem que auxilia os usuários na realização das atividades. Na área central são apresentadas as janelas do sistema.

Na Figura 28 são apresentadas as funcionalidades disponibilizadas em cada opção do menu. As funcionalidades correspondem aos casos de uso indicados na Seção 3.3.

Figura 28. Funcionalidades da ferramenta

Para o desenvolvimento das funcionalidades, buscou-se seguir o mesmo padrão para todas as telas, onde na parte superior são exibidos o título e a descrição dos procedimentos que devem ser realizados. Na parte inferior de todas as telas encontra-se o ícone (?) para obtenção de ajuda, sendo apresentado uma tela com informações pertinentes à cada operação que deve ser realizada. Também na parte inferior da tela são apresentados os botões que acionam as operações.

Na Figura 30 é apresentada a tela de definição do projeto, onde é possível informar o nome do projeto, localização do arquivo, autor e exercício que se pretende desenvolver. Todo projeto criado deve estar relacionado à um projeto.

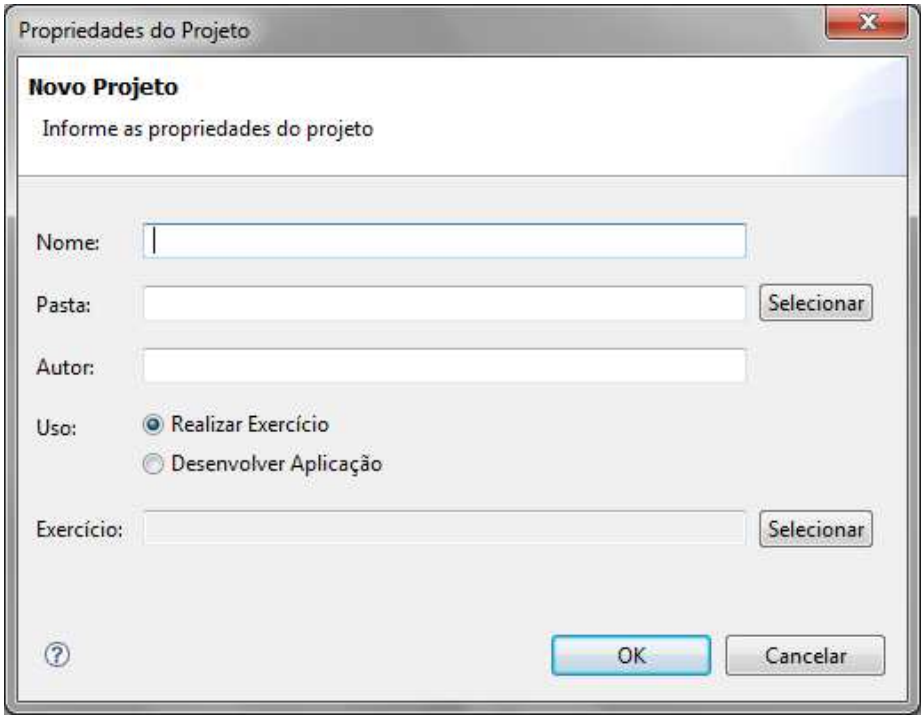


Figura 29. Tela de definição do projeto

Na Figura 30 é apresentada a tela de definição da fonte de dados, onde é possível definir as estruturas que irão armazenar as informações, bem como as configurações de acesso.

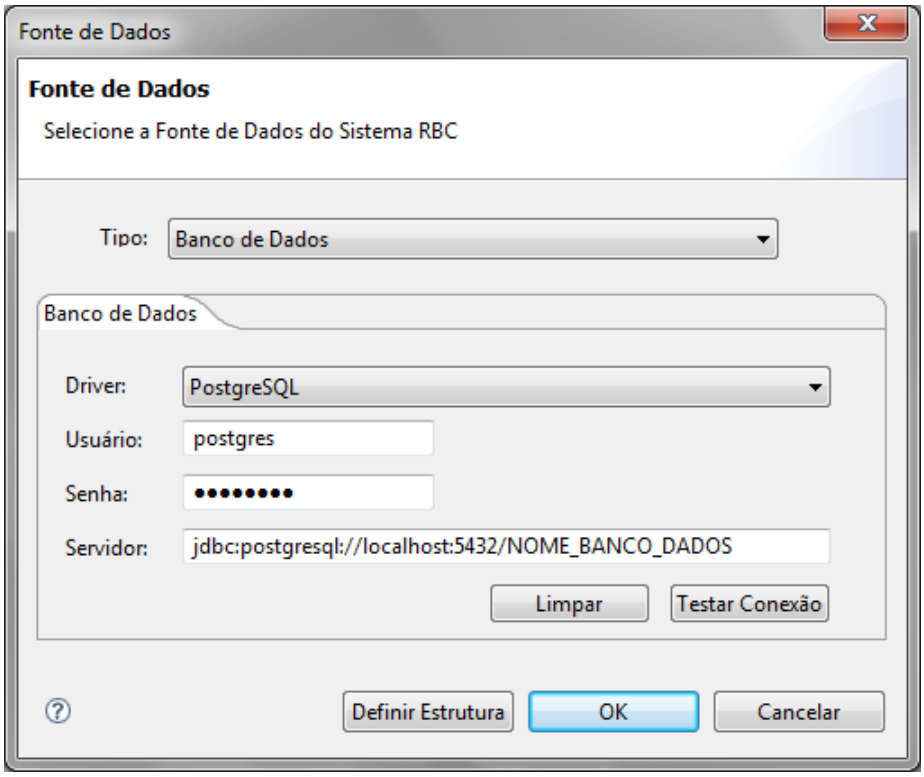


Figura 30. Tela de definição da fonte de dados

As duas formas de armazenamento e recuperação das informações são:

- Arquivo Texto: é solicitado ao usuário que informe o arquivo onde serão armazenadas as informações e o tipo de separador (*token*) que será utilizado entre os dados.

- Banco de Dados: conforme apresentado na Figura 30, são solicitadas as informações de configuração de acesso a base de dados, sendo que a ferramenta já disponibiliza as informações padrões, que são distribuídas pelos fabricantes dos bancos de dados. Estão disponíveis na ferramenta as configurações para os bancos de dados: Firebird[19], HSQLDB[20], MySQL[21], PostgreSQL[22] e SQLServer[23].

[19] Disponível em: http://www.firebirdsql.org/
[20] Disponível em: http://hsqldb.org/
[21] Disponível em: http://www.mysql.com/
[22] Disponível em: http://www.postgresql.org/
[23] Disponível em: http://www.microsoft.com/brasil/servidores/sql/default.mspx

Na Figura 31 são apresentadas as telas de definição das estruturas da fonte de dados e definição de campos da estrutura. O acesso a esta tela é realizado através da tela de definição da fonte de dados.

Na tela da Figura 31a é possível visualizar a estrutura de um banco de dados existente, onde o usuário pode definir através das caixas de checagem a esquerda dos campos, quais as informações serão utilizadas pelo sistema de RBC. A tela identifica os esquemas, tabelas, campos e relacionamentos do banco de dados. Também são identificados automaticamente os tipos de informação que cada campo armazena e transformados em tipos da linguagem de programação Java, para facilitar o uso dentro da ferramenta.

Na tela da Figura 31b é possível alterar ou cadastrar um novo campo da estrutura de dados existente. A ferramenta possibilita capturar a estrutura existente de uma fonte de dados, onde são mapeados os metadados de um banco de dados relacional ou identificados campos da estrutura de um arquivo TXT. Todas as alterações feitas nesta tela são diretamente executadas na estrutura, o que requer a atenção dos usuários. Na tela da Figura 31a o usuário pode clicar com botão direito do mouse sobre um determinado campo para cadastrar, alterar ou excluir informações.

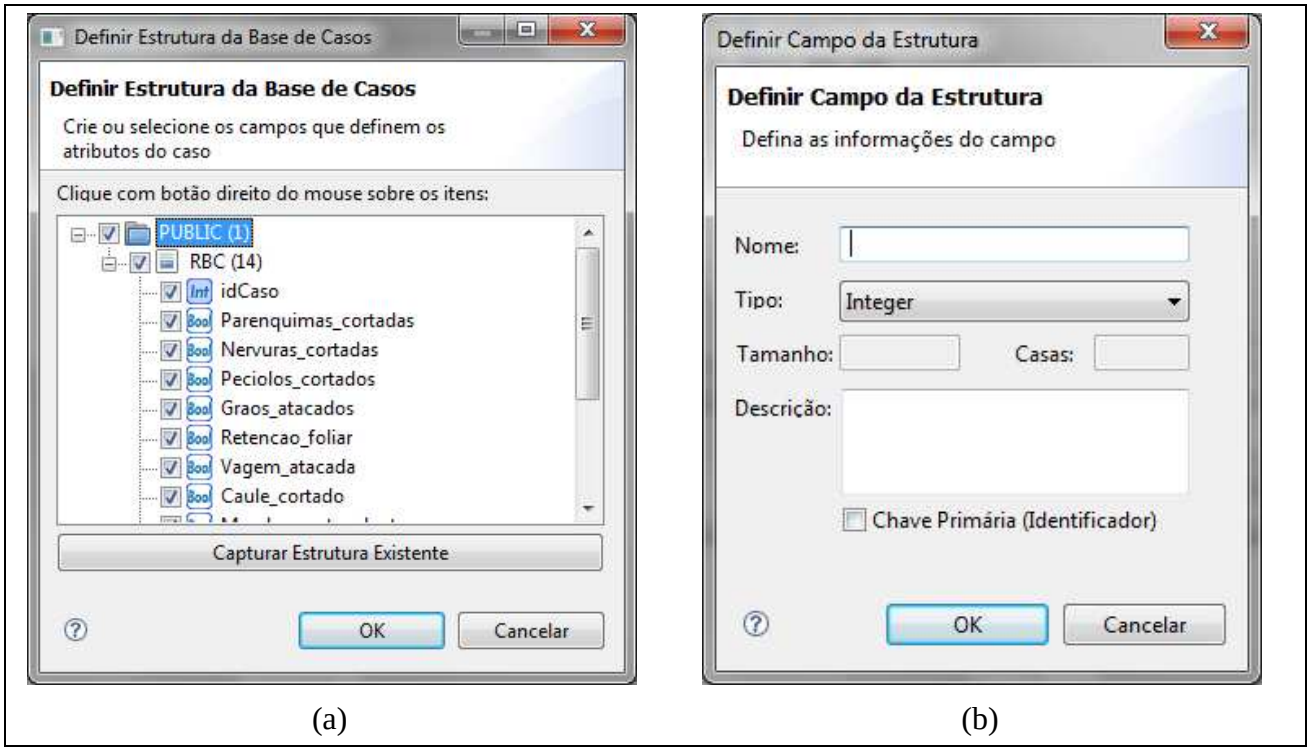


Figura 31. Tela de captura de estruturas existentes

Caso o usuário não informe uma fonte de dados existente, poderá definir os campos da estrutura utilizando a mesma funcionalidade. Ao concluir a definição (Clicando no botão OK) o sistema realiza a validação da estrutura, que avalia se os atributos informados são válidos e se as estruturas selecionadas estão relacionadas, no caso de selecionar duas ou mais tabelas de um banco de dados relacional. Nesta situação as tabelas devem possuir as referencias cadastradas (chaves estrangeiras).

Na Figura 32 é exibida a tela que realiza a definição da estrutura do caso, onde são definidos quais os atributos relacionados à definição do problema, a solução do problema e o resultado que obteve-se na solução do caso. Os atributos apresentados correspondem aos campos selecionados na tela de Definição da Estrutura da Base de Casos, sendo que os atributos selecionados são inicialmente definidos como Descrição.

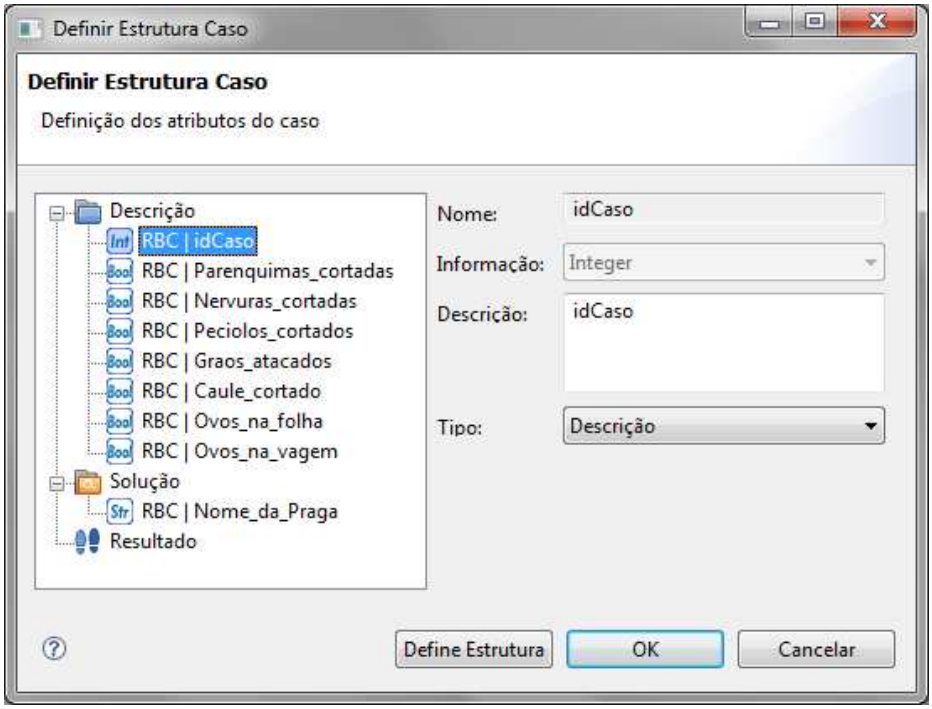


Figura 32. Tela de definição da estrutura do caso

Nas Figura 33 e Figura 34 são apresentadas as telas de definição de similaridade, onde na aba similaridade global pode-se definir uma das métricas disponíveis, o número de casos que são apresentados na recuperação, o percentual de similaridade mínimo para apresentação (limiar) e o tipo de definição de pesos. O usuário pode definir entre manter os pesos fixos nas configurações da aplicação, definir os pesos durante a execução do ciclo RBC ou não utilizar pesos.

Como similaridade global a ferramenta disponibiliza as métricas: a) Vizinho Mais Próximo Ponderado; b) Distância Euclidiana; c) Distância de Manhattan; d) Distância de ChebyChev; e) Distância de Casamento Simples.

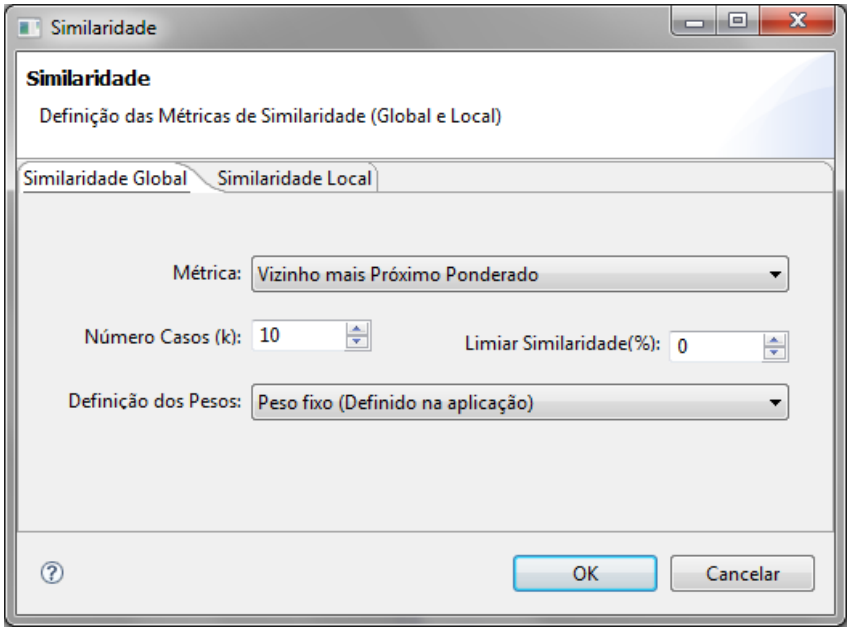


Figura 33. Tela de definição da similaridade global

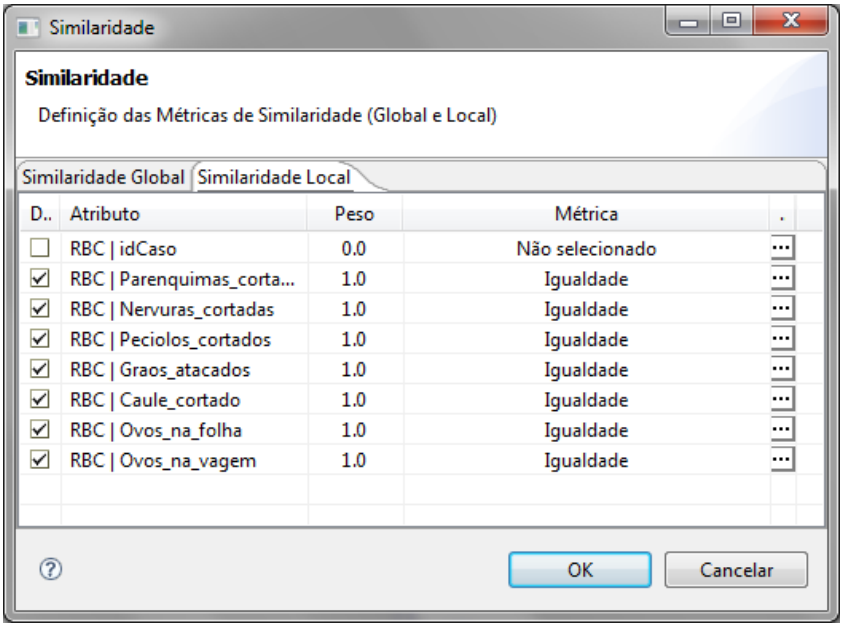


Figura 34. Tela de definição da similaridade local

Na aba similaridade local (Figura 34) é possível definir como será realizado o cálculo de similaridade entre cada atributo, definindo se o atributo é discriminante [24]e o peso que o atributo terá em relação aos demais (caso selecionada a opção Peso fixo na similaridade global). Também nesta tela é definida a métrica de similaridade local para cada atributo, sendo disponibilizado: a) Igualdade; b) Intervalo; c) Degrau; d) Função (Linear, Quadrática ou Normal); e) Matriz de Similaridade; f) Categorias Ordenadas; g) Maior Seqüência de Caracteres; e g) Textual. Para a métrica Textual foi utilizada a biblioteca extensível de métricas de similaridade SimMetrics[25], que é disponibilizada como software livre. Da biblioteca foram incluídos os algoritmos (definido como parâmetro desta métrica): a) BlockDistance; b) CosineSimilarity; c) EuclideanDistance; d) JaccardSimilarity; e) Jaro; f) QGramsDistance; e g) NeedlemanWunch.

As métricas de similaridade local retornam valores no intervalo entre 0 e 1, onde 1 representam valores totalmente similares e 0 representam valores totalmente distintos. Para cada métrica de similaridade local o usuário deve definir os parâmetros, sendo apresentada uma tela específica para cada métrica, exibidas ao acionar o botão ao lado do nome da métrica (...).Os parâmetros de similaridade de cada métrica são:

- Igualdade: pode-se definir a negação da igualdade, onde caso os valores atributos comparados forem diferentes, a função retorna similaridade 1. Também pode-se definir para atributos literais a diferenciação entre caracteres maiúsculos e minúsculos.

- Intervalo: defini-se o valor de intervalo que é considerado na comparação. A similaridade entre os atributos é considerada verificando se a diferença dos valores encontra-se dentro do intervalo, levando em conta os limites determinados. Pode-se optar por considerar apenas diferenças maiores ou menores que zero na comparação.

- Degrau: define-se o valor de *threshold*,(limiar) e a opção de definir como similar a diferença Maior ou Menor.

[24] Atributos que são utilizados para realizar a comparação entre o caso e a situação presente no cálculo da similaridade.
[25] Disponível em http://www.dcs.shef.ac.uk/~sam/simmetrics.html. Acessado em 01/11/2009.

- Função: define-se inicialmente a função que deseja-se aplicar (Linear, Quadrática ou Normal) e os parâmetros das funções. Também deve-se informar os valores mínimos e máximos que podem ser utilizados, para que seja possível definir um valor de similaridade no intervalo esperado. A partir destas definições pode-se visualizar o gráfico da opção desejada na tela.

- Matriz de Similaridade: definem-se as categorias que compõe a matriz e também os pesos das relações entre as categorias. Pode-se optar por utilizar uma matriz simétrica (valores acima da diagonal superior da matriz são iguais da inferior) ou assimétrica (valores da matriz podem ser diferentes).

- Categorias Ordenadas: definem-se as categorias, e também tem-se a opção Cíclica, onde durante a comparação do caso atual e o caso da base, são comparadas a primeira e a última categoria, elas terão similaridade próxima de 1, e caso não seleciona a opção, serão totalmente diferentes.

- Maior Seqüência de Caracteres: está métrica não possui parâmetros à definir, porém exibe uma tela mostrando seu funcionamento. Nela são comparadas cadeias de caracteres similares entre os casos.

- Textual: define-se o algoritmo de comparação e também se será considerada a diferenciação entre caracteres maiúsculos e minúsculos. Na tela de definição pode-se realizar o cálculo de similaridade entre duas sentenças de caracteres informadas pelo usuário, possibilitando testar o funcionamento das funções.

Exemplos de telas de definição dos parâmetros são apresentados na Figura 35a (Métrica Intervalo) e Figura 35b (Matriz de Similaridade).

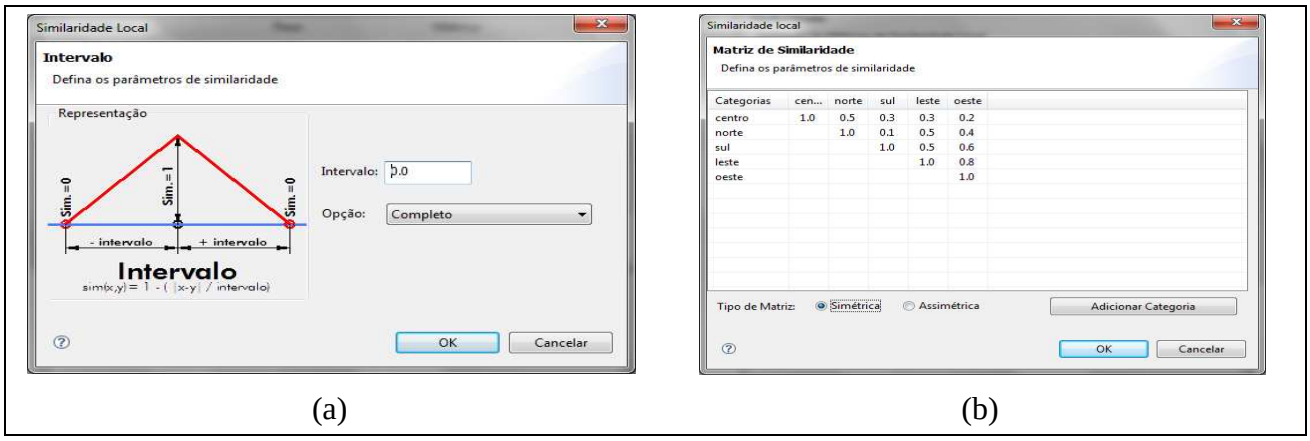


Figura 35. Telas de definição dos parâmetros de similaridade

Observa-se que cada métrica de similaridade possui aplicação em determinados tipos de atributo (Literais, Inteiros, etc). Para auxiliar o usuário a ferramenta disponibiliza na ajuda uma tabela descrevendo as possibilidades de métricas para cada tipo de atributo. Além disso é realizada a verificação no momento da definição da métrica. Também é avaliado se os parâmetros foram informados e se os pesos, caso sejam informados na aplicação, são válidos (maiores que 0).

Outra definição realizada na ferramenta é Adaptação, onde o usuário pode selecionar os atributos do caso que irão ser ajustados durante o ciclo RBC. A tela de adaptação é exibida na Figura 36, onde o usuário pode selecionar o atributo que sofrerá a adaptação e o campo que irá ser aplicado na adaptação. Os métodos disponíveis são Copia do Atributo, Proporção Numérica Direta, ou Adaptação Nula, caso não seja realizado nenhuma adaptação. Os atributos que sofrem adaptação ficam com ícone marcado na cor vermelha na listagem.

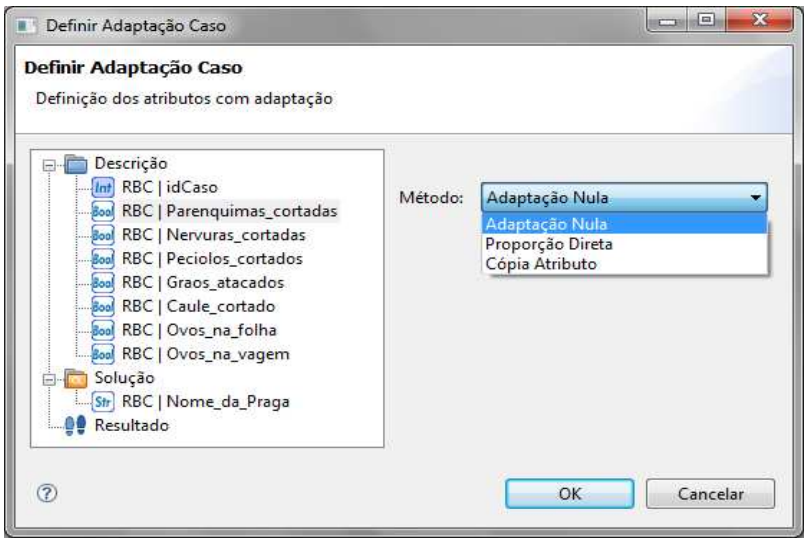


Figura 36. Tela de definição da adaptação

Após a definição do sistema de RBC o usuário pode realizar o ciclo RBC, onde se executa cada etapa e se visualiza os respectivos resultados. Na Figura 37 é exibida a tela de Execução do Ciclo RBC. Ao iniciar esta funcionalidade a aplicação realiza a geração das classes e arquivos de configuração da aplicação e em seguida os executa. Caso ocorra algum problema nas definições o sistema apresenta uma mensagem de erro, descrevendo o procedimento que deve ser realizado para solucionar o problema.

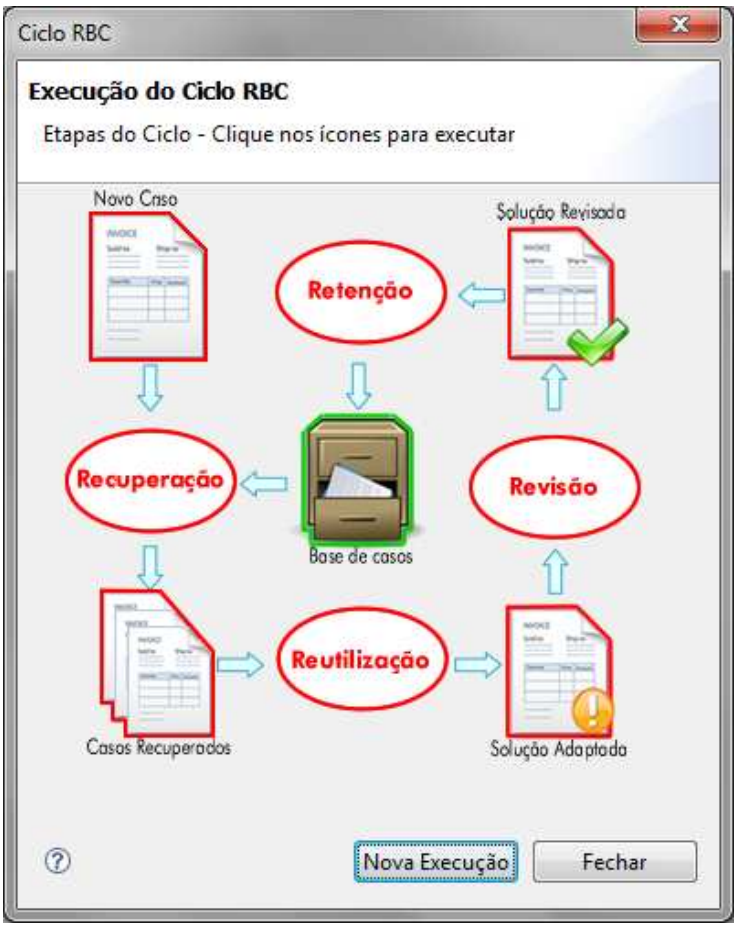


Figura 37. Tela de Execução do ciclo RBC (Ciclo 4R's)

Nesta tela cada ícone representa uma etapa do ciclo RBC e para executar deve-se clicar sobre as imagens. No centro encontra-se a base de casos, onde as informações do sistema ficam armazenadas e podem ser consultadas. O ciclo inicia pela definição de um novo caso. As setas indicam o fluxo do ciclo.

Na tela de definição do novo caso, o usuário deve informar os valores dos atributos discriminantes. Caso seja definido que os pesos serão informados pelo usuário durante a execução, também é apresentado ao usuário, ao lado de cada campo, um componente para definir o peso do

atributo em relação aos demais. Ao informar os valores e confirmar, a primeira etapa do ciclo é concluída, e então fica marcada na cor Verde identificando que já foi realizada.

Seguindo o fluxo, o usuário deve clicar na etapa de recuperação, onde o sistema faz o cálculo de similaridade recuperando os casos mais similares da base. Caso tenha recuperado um ou mais casos o sistema marca esta etapa e a seguinte (Casos Recuperados) como concluída. Na Etapa Casos Recuperados o usuário pode visualizar os casos recuperados com seus respectivos valores de similaridade, conforme apresentado na Figura 38. Caso não tenha encontrado nenhum caso similar (base de casos vazia ou com filtros iniciais que não satisfazem as condições), o sistema informa ao usuário que não foi localizado nenhum caso similar e pergunta se deseja armazenar o caso atual na base. Caso confirme o ciclo segue para etapa de Revisão.

Na Etapa de Reutilização (Adaptação), que é opcional no sistema de RBC, o sistema realiza a adaptação dos atributos dos casos recuperados na etapa anterior e possibilita a visualização através da etapa Solução Adaptada. Nesta opção o usuário visualiza os casos adaptados e deve selecionar o caso mais similar para ser realizada a próxima etapa do ciclo.

Na etapa de Revisão as informações do caso mais similar são revisadas, e realizada a adequação caso seja necessária. Nesta opção o sistema pergunta se deseja copiar os dados informados na situação atual para o caso selecionado como mais similar. Caso confirme, as informações são substituídas no caso selecionado. O usuário pode visualizar os dados do caso e, caso necessário, pode realizar alterações nos valores para adequar o caso à situação real.

Após a revisão o usuário pode visualizar os dados informados na etapa de Solução Revisada e caso estejam todos os dados corretos pode executar a etapa de Retenção, onde o novo caso é armazenado na base de casos para ser utilizado em situações futuras.

O sistema não permite executar etapas fora da seqüência do fluxo, com exceção de quando a execução do ciclo RBC não localizar nenhum caso similar, passando desta forma da etapa de recuperação para revisão, se o usuário optar por armazenar o novo caso. As etapas já realizadas podem ser consultadas ou alteradas, devendo o usuário seguir as etapas seguintes à alterada para concluir o ciclo. O botão Nova Execução permite a qualquer momento reiniciar o ciclo.

Na Figura 38 é apresentada a tela de visualização da base de casos, onde todos os casos podem ser consultados. Para cada atributo a ferramenta exibe uma coluna, que pode ser redimensionada, indicando com ícones diferenciados os atributos que correspondem à descrição,

solução e resultado do caso. Caso esteja sendo executado o ciclo RBC, a ferramenta apresenta na primeira coluna o valor de similaridade do caso armazenado em relação ao caso atual. Caso o usuário deseje visualizar as informações com mais detalhes, pode clicar duas vezes sobre o caso para que seja exibida uma tela com apenas os dados do caso selecionado.

Ao acessar a base de casos pela opção do Menu Execução, o usuário pode realizar a manutenção da base de casos, possibilitando adicionar e remover casos. As informações de cada caso são apresentadas nas linhas da listagem.

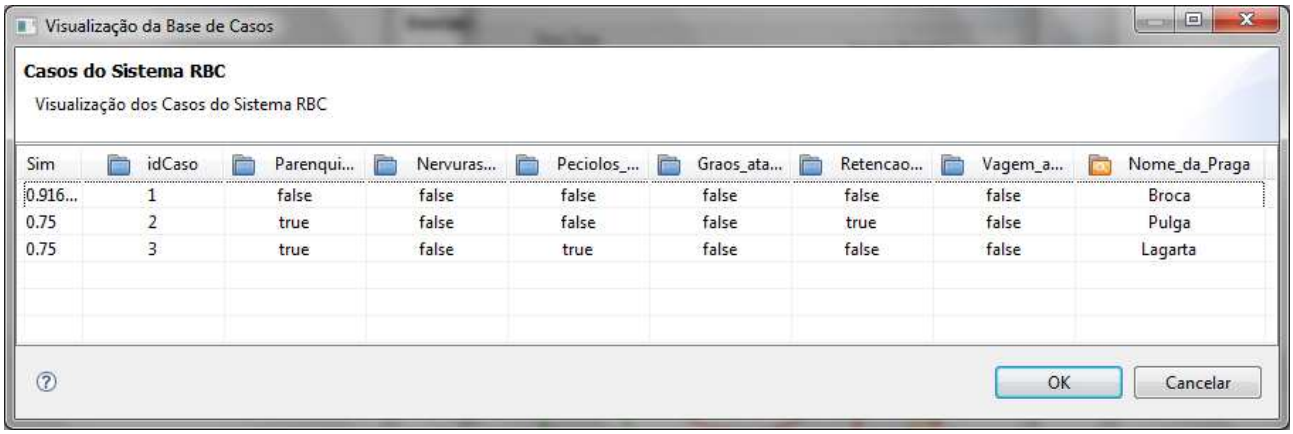


Figura 38. Tela de Visualização da Base de Casos

Para o ensino da técnica de RBC foram criadas duas funcionalidades, uma área específica para tutores criarem exercícios e outra para os alunos desenvolverem estes exercícios. Para criar um exercício o usuário (Tutor) deve informar a descrição do problema, as estruturas esperadas na resolução do problema e também as possíveis dicas que podem ser consultadas pelos alunos durante a resolução do exercício. Na Figura 39 é apresentada a tela de criação de exercício.

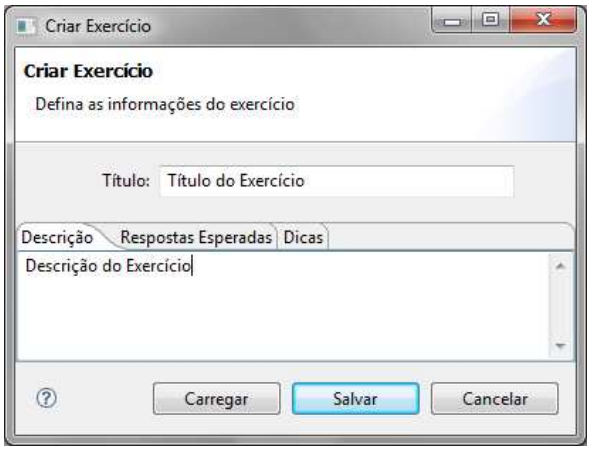
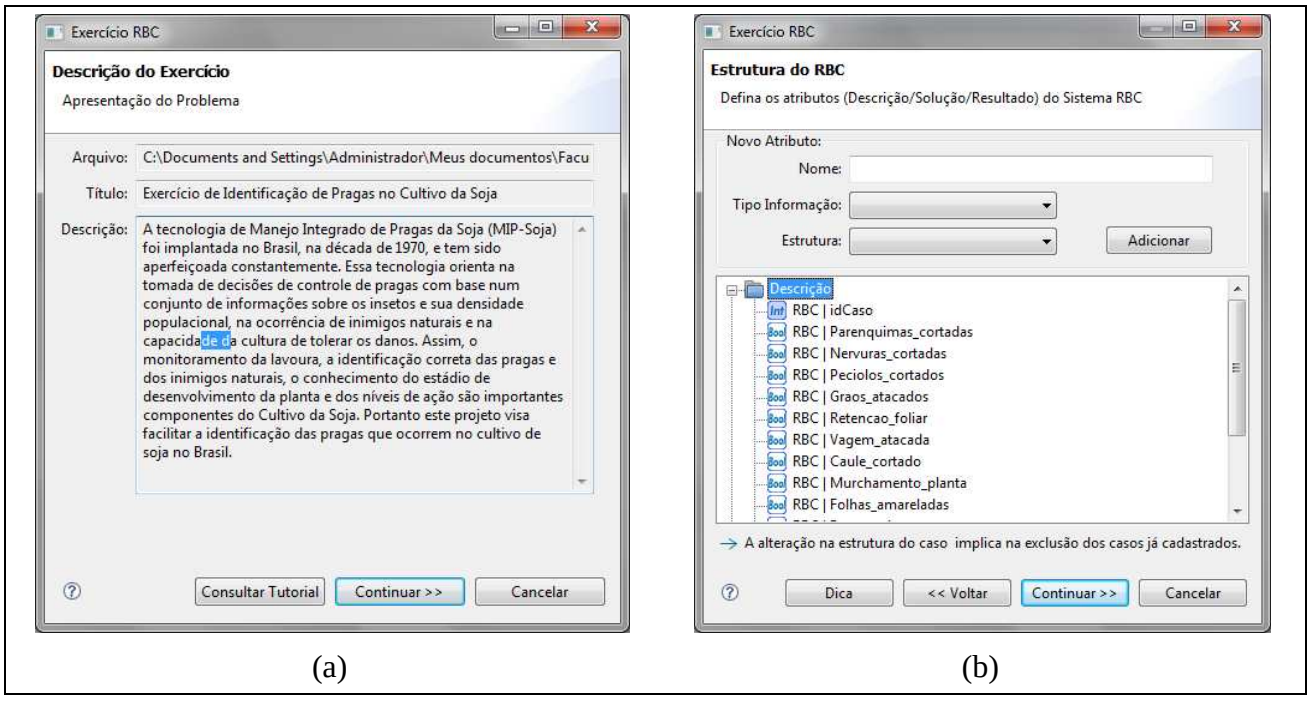


Figura 39. Tela de criação de exercício

Para desenvolver o exercício proposto pelo tutor, o usuário (aluno) deve iniciar um novo projeto e informar na definição o arquivo referente ao exercício. Ao acessar a funcionalidade Realizar Exercício o usuário é apresentado à descrição do exercício (Etapa de Apresentação do Problema da PBL), conforme apresentado na Figura 40a. Nesta tela pode-se optar por consultar o tutorial para visualizar informações pertinentes aos conceitos de RBC, e também, após obter as informações relevantes da descrição, pode continuar seguindo para tela de Estrutura do RBC (Figura 40b).



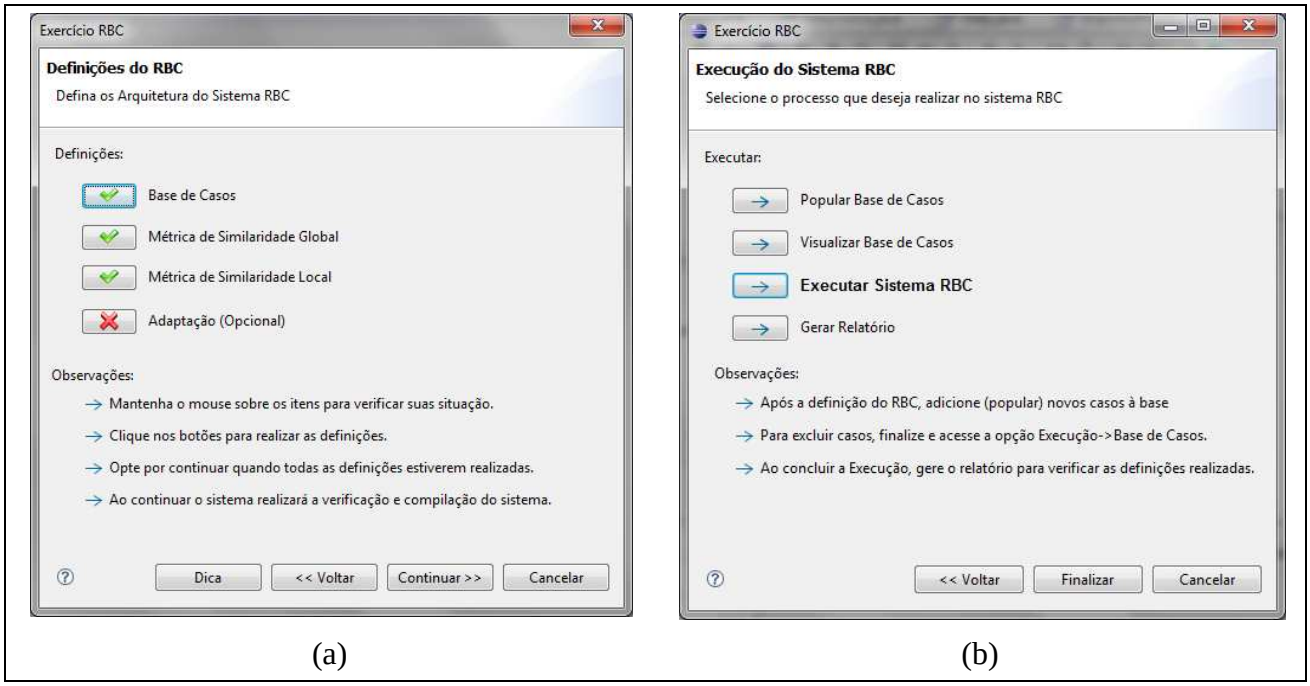
(a) (b)

Figura 40. Telas de descrição do exercício e definição da estrutura.

Na tela de estrutura do RBC (Etapa de Assimilação da PBL) o usuário deve informar os atributos que identificou na descrição do problema, indicando o tipo de informações que este atributo irá armazenar e também de qual estrutura o atributo pertence (Descrição, Solução ou Resultado). Pode-se consultar as dicas para auxiliar nas definições ou ainda voltar para visualizar a descrição do problema. Após esta definição o usuário opta por continuar, onde então o sistema valida a estrutura e gera uma fonte de dados com estes metadados para armazenar as informações.

A estrutura é gerada através de um Banco de Dados Relacional HSQLDB, que pode ser embarcado na aplicação e não necessita ter um servidor para manipulação dos dados. Caso não seja validada a estrutura ou não seja possível criar a base de dados, o sistema apresenta uma mensagem de erro, indicando a possível solução do problema.

Ao continuar, o usuário é direcionado para tela de Definição da Estrutura, conforme apresentada na Figura 41a (Etapa de Resolução do Problema).



(a) (b)

Figura 41. Telas de definição e execução.

Na tela da Figura 41a são feitas as definições do sistema de RBC, devendo o usuário clicar sobre os botões com ícones para carregar as funcionalidades. Os botões indicam se a definição foi realizada de maneira correta. Caso o mouse seja mantido sobre o botão, é exibida uma janela com as informações da situação atual, sendo que ao realizar cada definição o sistema valida e atualiza os dados.

Continuando a execução, o sistema gera os arquivos da aplicação e passa para tela de Execução (Figura 41b, correspondente à Validação dos Resultados da PBL), onde realiza-se testes para analisar o funcionamento do sistema desenvolvido. Aqui o usuário pode Popular a Base de Casos, adicionando casos manualmente ao sistema, pois como o sistema foi desenvolvido sua base de casos estará vazia e o sistema de RBC não funcionará de maneira adequada sem casos cadastrados. Na opção Visualizar Base de Casos pode-se visualizar os casos já cadastrados e na opção Executar Sistema RBC, pode-se executar o sistema seguindo o ciclo RBC.

Após a realização dos testes o usuário pode acessar a opção Gerar Relatório, onde são apresentadas todas as definições realizadas no ambiente, para o tutor analisar o desempenho do

aluno na resolução do exercício (Etapa de Avaliação Final da PBL). Na Figura 42 é apresentada a tela de visualização do relatório, onde pode-se salvar ou imprimir as informações.

Figura 42. Telas de visualização do relatório

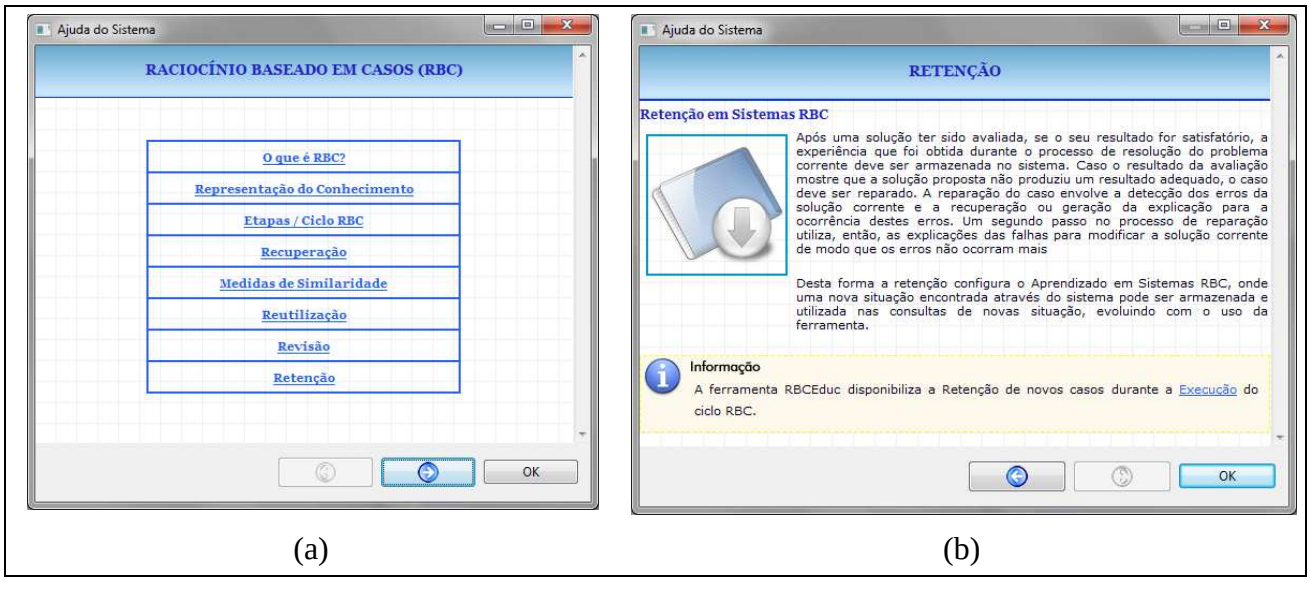
No relatório são apresentadas as seguintes informações:

- Propriedades do Projeto: nome do projeto, autor, data de criação e se foram realizados testes na ferramenta.
- Exercício Relacionado: a descrição do exercício.
- Fonte de Dados: informações da fonte de dados que armazena as informações.
- Base de Casos: método de recuperação e os filtros, caso tenham sido aplicados.
- Estrutura: atributos e tipos de informações que foram definidas.
- Similaridade: métricas de similaridade global e local, com os respectivos parâmetros definidos em cada métrica.
- Adaptação: métodos e atributos caso tenham sido definidos.
- Base de Casos: apresenta as informações cadastradas na base de casos.
- Estatísticas: apresenta o total de vezes que os alunos realizaram cada procedimento dentro da ferramenta, possibilitando avaliar as principais dificuldades entradas pelos

alunos. Estão disponíveis o número de vezes que o aluno: a) Visualizou a descrição do Problema; b) Visualizou as dicas; c) Tentou gerar a estrutura; d) Gerou a estrutura com sucesso; e) Definiu a base de casos; f) Definiu a similaridade global; g) Definiu a similaridade local; h) Definiu a adaptação; i) Casos adicionados manualmente à base; j) Casos removidos da base; k) Visualizou a Base de Casos; l) Ciclos RBC iniciados; m) Ciclos RBC concluídos; e n) Emissões do Relatório.

Estas informações também podem ser visualizadas ao acessar a funcionalidade Imprimir. Um relatório completo é apresentado no Apêndice A. O relatório traz as definições e as principais operações executadas na ferramenta, não apresentando as execuções realizadas pelo usuário com os respectivos resultados. Para visualizar os resultados das execuções o usuário pode acessar diretamente a opção Executar RBC.

Na Figura 43a é apresenta a tela inicial do Tutorial RBC, onde se pode obter o embasamento teórico sobre a técnica de RBC. O Tutorial é composto por oito tópicos (Definição, Representação do Conhecimento, Ciclo RBC, Recuperação, Métricas de Similaridade, Reutilização, Revisão, Retenção), que descrevem os conceitos envolvidos e fazem a ligação com as definições que são realizadas na ferramenta utilizando *links*.



(a) (b)

Figura 43. Telas de tutorial e ajuda do sistema.

Na Figura 43b é visualizado o tópico Retenção com os respectivos conceitos relacionados e na parte inferior são apresentadas as informações de como realizar esta etapa na ferramenta, podendo o usuário clicar no link “Execução” para ir para tela de ajuda da ferramenta.

As ajudas da ferramenta podem ser acessadas a qualquer momento através da tecla de atalho F1 ou pelo botão “?”, localizado na parte inferior de cada tela. Cada tela possui uma ajuda relacionada, para que o usuário possa sempre obter auxílio na definição que está realizando. Foi criado um total de 36 arquivos de ajuda contextualizada, sendo no Apêndice B apresentada a estrutura (mapa) que mostra cada uma delas. Na Figura 45 é exibido um exemplo de ajuda, referente a tela de definição do projeto.

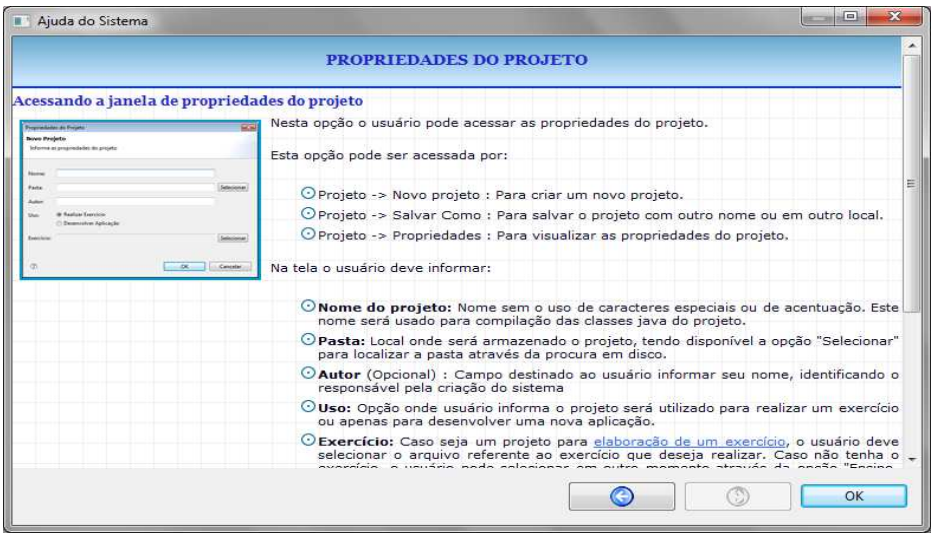


Figura 44. Telas de tutorial e ajuda do sistema.

Tanto nas ajudas como no tutorial é possível navegar através dos *links* disponíveis, de maneira similar à *websites* da internet, que fazem as ligações entre os conceitos e funcionalidades da ferramenta.

Além das definições e execuções que podem ser realizadas dentro da ferramenta, pode-se também exportar a aplicação desenvolvida através da funcionalidade Exportar. Nesta opção a ferramenta gera os arquivos das classes e configurações e armazena na pasta do projeto, possibilitando que o sistema desenvolvido seja executado fora da ferramenta. A interface disponibilizada pela ferramenta é através do modo Console (linha de comando), porém os usuários podem desenvolver interfaces gráficas ou ainda utilizar o código fonte em outras aplicações. No Apêndice C é apresentado um exemplo de código fonte gerado pela ferramenta.

Com a conclusão do desenvolvimento foram realizados testes, buscando melhorar o comportamento da ferramenta. Após a manutenção e melhoria de algumas funcionalidades foi possível realizar a validação com alunos da disciplina de Inteligência Artificial de um Curso de Ciência da Computação, conforme apresentado no Capítulo 4, Resultados.

4 RESULTADOS

Com o desenvolvimento da dissertação obteve-se a fundamentação teórica da técnica de Raciocínio Baseado em Casos, com detalhamento dos conceitos e uma análise das soluções existentes para desenvolvimento de sistemas utilizando esta técnica. Este material serviu como base para o projeto e modelagem da solução, gerando artefatos que auxiliaram na programação.

O desenvolvimento seguiu as etapas planejadas no início do projeto, e teve como produto a ferramenta que possibilita à usuários desenvolver de forma simples e ágil novos sistemas de RBC, como também exercitar os conceitos envolvidos, sem se preocupar com detalhes específicos de programação.

A ferramenta possibilita professores de Cursos de Graduação, com aplicação na área de Inteligência Artificial, utilizar para o ensino da técnica desenvolvendo exercícios com o objetivo de abordar os conceitos envolvidos na técnica de RBC. Além do ensino é possível desenvolver aplicações para serem utilizadas junto à sistemas existentes.

Para avaliar o desempenho no ensino da técnica, foi disponibilizada a alunos do curso de graduação em Ciência da Computação, junto à disciplina de Inteligência Artificial, numa turma com 12 alunos. Como definição do cenário de testes, foi solicitado pelo professor da disciplina que os alunos elaborassem problemas que pudessem ser resolvidos com o uso da técnica de RBC. Estes alunos não haviam desenvolvido nenhum sistema de RBC em atividades anteriores, apenas tinham o embasamento teórico básico sobre a técnica.

Assim, a descrição dos problemas elaborados pelos próprios alunos foi revisada pelo professor que selecionou as mais pertinentes para que fossem desenvolvidas pelos alunos. Desta forma, cada problema foi desenvolvido por 2 alunos. Os tipos de problemas levantados foram soluções para *help desk*, auxílio na identificação de pragas no cultivo da soja, diagnóstico de problemas em veículos, sistema de recomendação para compra de jogos, sistema de apoio à decisão em jurisprudência, sistema de recomendação para compra de computadores, e sistema de auxílio à escolha da profissão.

A ferramenta foi disponibilizada aos alunos, onde inicialmente foram apresentadas as principais funcionalidades disponíveis. A realização do exercício não foi supervisionada e os alunos

ao fim responderam à um questionário de avaliação (Apêndice D) , que serviu como referência para melhorias na ferramenta.

O questionário possui duas vertentes: avaliação da usabilidade e da funcionalidade. O questionário avalia a percepção do usuário após o uso da ferramenta, tendo como escala: a) Extremamente; b) Razoavelmente; c)Ligeiramente; d)Neutro/Não sei; e e)Nenhum Pouco/Não.

A avaliação da usabilidade foi baseada no trabalho de Gladcheff (2000), intitulado “Um instrumento de avaliação da qualidade para software educacional de Matemática”. que por sua vez é baseada nas heurísticas de Nielsen (1992), que descrevem propriedades que devem ocorrer em sistema com alta usabilidade, possuindo um total 18 questões objetivas.

A avaliação funcional abordou questões de uso das funcionalidades da ferramenta, possuindo um total de 10 questões objetivas e 2 questões discursivas, onde o usuário informou as dificuldades encontradas e o diferencial identificado na ferramenta.

Para apresentação da avaliação as respostas foram transformadas de valores qualitativos para quantitativos, definindo uma escala de 0 a 10, sendo 0 o termo Nenhum Pouco/Não e 10 para o termo Extremamente, ficando os demais distribuídos no intervalo.

Os valores apresentados nas avaliações correspondem a todas as respostas que os alunos forneceram, mesmo os que não conseguiram executar todo o processo de desenvolvimento. A descrição (enunciado) dos quesitos é apresentada no formulário de avaliação (Apêndice D) e um resumo destas é apresentado nos rótulos dos gráficos apresentados.

No dia da avaliação alguns problemas foram encontrados pelos alunos, sendo estes não identificados nos testes iniciais durante o desenvolvimento. Um dos problemas foi que os alunos procuravam iniciar a definição do sistema RBC utilizando um banco de dados inexistente, sem utilizar a funcionalidade de realizar exercício, que não necessita desta configuração, outro problema foi pela utilização de caracteres especiais no nome do projeto e dos atributos. Tais problemas limitaram para alguns alunos o uso, por não conseguirem identificá-los, porém depois de identificados e esclarecidos como resolver foi possível continuar o exercício sem necessitar de ajustes.

Na Figura 45 são apresentadas as médias e o desvio padrão da avaliação da usabilidade.

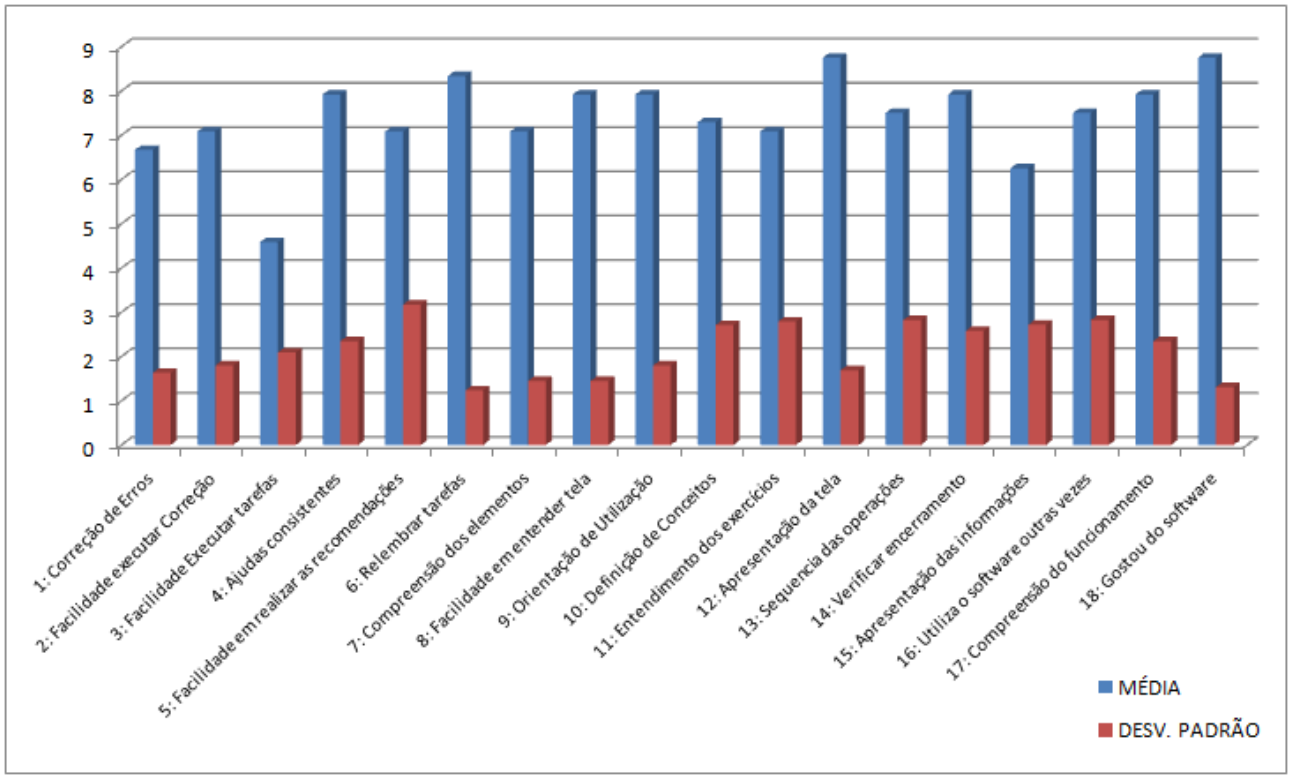


Figura 45. Gráfico dos quesitos de usabilidade.

Para os 18 quesitos de usabilidade avaliados, a média geral foi de 7,42, com desvio padrão de 2,34, variando a média entre 4,58 e 8,75, e desvio padrão entre 2,82 e 1,23.

Alguns quesitos chamam a atenção como o 3: Facilidade de executar tarefas, que ficou com média 4,58 e desvio padrão de 1,79. A dificuldade apresentada pelos alunos foi de encontrar onde estava localizada a opção de Iniciar a Realização de um Exercício. Desta forma estes alunos com dificuldade tentaram desenvolver o exercício sem utilizar a funcionalidade específica para este fim, definindo uma fonte de dados (servidor de banco de dados) que não estava disponível, o que não possibilitou continuarem o desenvolvimento além da definição da fonte de dados.

Buscando solucionar esta deficiência foi alterada a forma como o usuário pode acessar a funcionalidade de exercício, sendo definido agora ao criar o projeto. Ao confirmar a criação do projeto o sistema pergunta ao usuário se deseja iniciar a realização do exercício. Também foi alterada a localização da opção no menu, onde se criou um menu Ensino, ao lado do menu Projeto, onde tem-se a opção de Visualizar o Tutorial e Realizar Exercício.

O principal problema identificado foi a possibilidade de os alunos informarem no nome dos atributos do caso com o uso de caracteres especiais (espaços, barras, pontuação entre outros). Este

problema limitou o uso para alguns alunos e não possibilitou que seguissem para as próximas etapas de definição do exercício. Como para cada aplicação são gerados os códigos fonte e os arquivos de configurações, o nome dos atributos é utilizado como variáveis nas classes do projeto, o que não possibilitou a compilação e execução.

Para solucionar esta deficiência foi bloqueada a digitação de caracteres especiais no nome dos atributos e reformulada a maioria das mensagens de erro, buscando identificar melhor os erros encontrados e descrevendo claramente as possíveis soluções. Isso afetou diretamente nos quesitos 1: Correção de Erros, que obteve média 6,67 e desvio padrão de 1,63, e o quesito 2: Facilidades de Executar Correções, que obteve média 7,08 e desvio padrão de 1,79.

No quesito 15: Apresentação das Informações, que obteve média 6,25 e desvio padrão de 2,72 os alunos indicaram como ponto fraco o tamanho das telas que não poderia ser redimensionada, e em algumas situações, dificultava a visualização das informações. Desta forma em algumas telas foi aumentado seu tamanho, como no caso da visualização da base de casos, e também se permitiu que algumas telas fossem redimensionadas (maximizadas) para melhor visualizar as informações.

Para os quesitos 7: Compreensão dos elementos (Média 7,08 e Desvio Padrão 1,44) e 10: Definição de Conceitos (Média 7,29 e Desvio Padrão 2,71), as telas de realização dos exercícios foram melhoradas incluindo comentários sobre o que deve ser realizado em cada etapa.

Os demais quesitos foram observados e considerados satisfatórios na avaliação, porém alguns ajustes nas ajudas foram realizados para esclarecer dúvidas que surgiram dos alunos.

Na Figura 46 são apresentadas as médias e o desvios padrão da avaliação de funcionalidade.

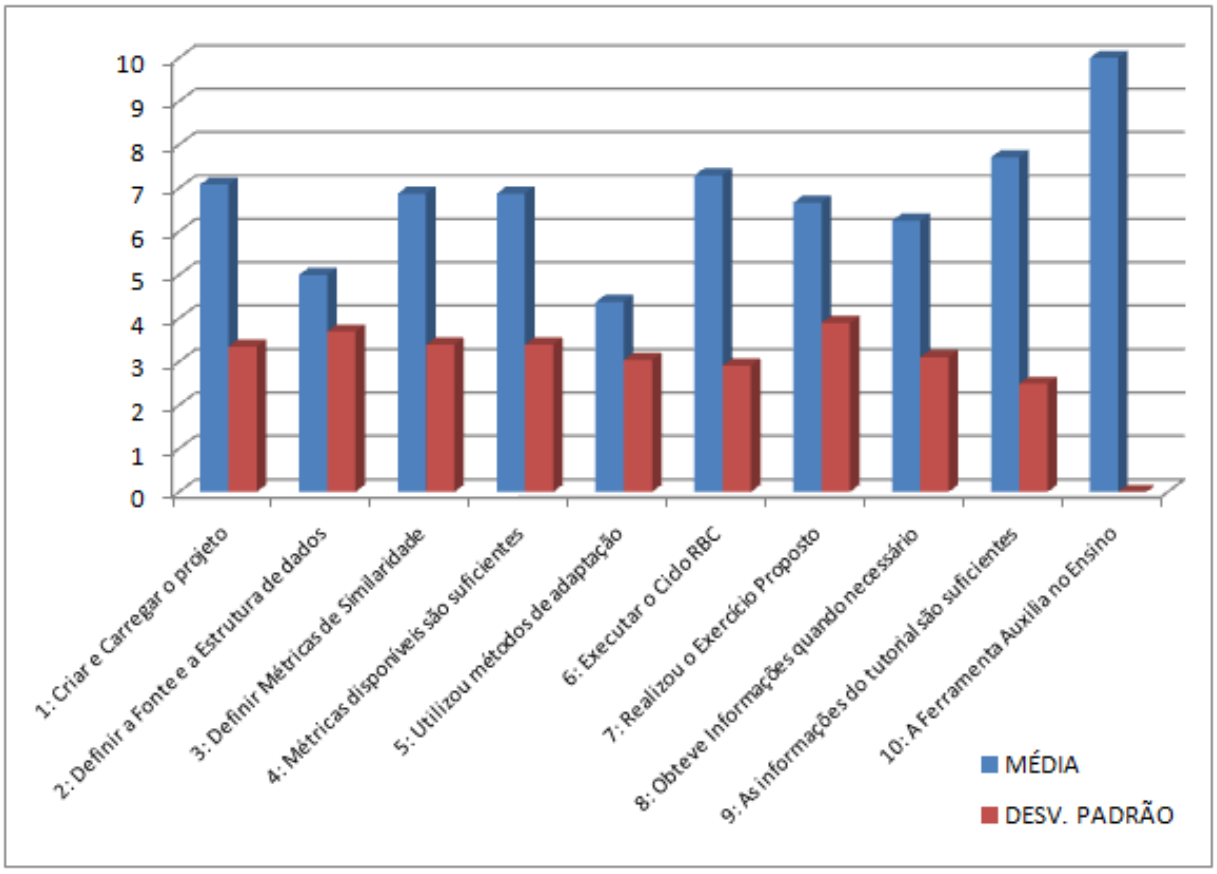


Figura 46. Gráfico dos quesitos de funcionalidade.

Para os 10 quesitos de funcionalidade avaliados, a média geral foi de 6,81 com desvio padrão de 3,32, variando a média entre 4,38 e 7,71, e desvio padrão variando entre 0 e 3,89. Observa-se que o desvio padrão foi alto, isso em decorrência de alguns alunos (4 no total) não conseguirem desenvolver o exercício sem auxílio do desenvolvedor, avaliando os quesitos de forma insatisfatória.

O grande problema encontrado foi na definição do nome dos atributos, como descrito acima, onde foi realizada a correção na ferramenta. A grande melhoria realizada quanto à estes quesitos foi nas mensagens de erro, onde foram ajustadas as mensagens e as respectivas correções necessárias.

O quesito 5: Utilizou Métodos de adaptação, que obteve média 4,38 e desvio padrão 3,04, não mostrou valores satisfatórios em decorrência de alguns alunos não realizarem adaptação nos exercícios desenvolvidos, definindo como resposta a opção Neutro/Não sei.

O quesito 2: Fonte de Dados, que obteve média 5,00 e desvio padrão de 3,69, foi indicado por alunos que não localizaram a funcionalidade de realização de exercício e não conseguiram efetivamente realizar a definição da fonte de dados.

No quesito 3: Definir Métrica de Similaridade, que obteve média 6,88 e desvio padrão de 3,39, foi identificado um problema que ao definir um limiar de similaridade mínimo para exibição dos casos, o sistema não apresentava nenhum caso, o que já foi corrigido na versão atual.

Uma melhoria que foi realizada, e que afeta a maioria dos quesitos, foi o ajuste das informações existentes nas ajudas, onde foram adicionadas informações que os alunos não obtiveram resposta quando precisaram.

Outra observação identificada com os testes foi ao adicionar novos casos, onde não era verificado se casos com as mesmas informações já estavam cadastrados, fazendo com que houvesse casos redundantes na Base de Casos. Desta forma foi realizada a alteração para que antes de armazenar um novo caso seja verificado se já existem casos idênticos, e se existirem, solicite a confirmação do usuário para cadastrar o caso novamente, minimizando desta forma a inconsistência na base de casos.

O principal quesito que não pode deixar de ser comentado foi o 10: A ferramenta Auxilia no Ensino, que obteve a média 10, comprovando que na opinião dos alunos pode auxiliar no ensino da técnica de RBC, sendo este indicado como principal diferencial.

De maneira geral a avaliação conseguiu identificar os principais pontos de melhoria, sendo estes apresentados e ajustados. Após a avaliação, com auxílio do professor e do mestrando, todos os alunos conseguiram desenvolver e testar os sistemas de RBC propostos. Do total de 12 alunos, 04 precisam de auxílio para o desenvolvimento do trabalho.

Como forma de visualizar o desempenho dos alunos na realização dos exercícios, é possível visualizar as estatísticas apresentadas no Relatório do Projeto, onde se tem contabilizado o número de vezes que cada funcionalidade foi executada durante a realização do exercício. A Figura 47 apresenta o gráfico da média e do desvio padrão destes valores.

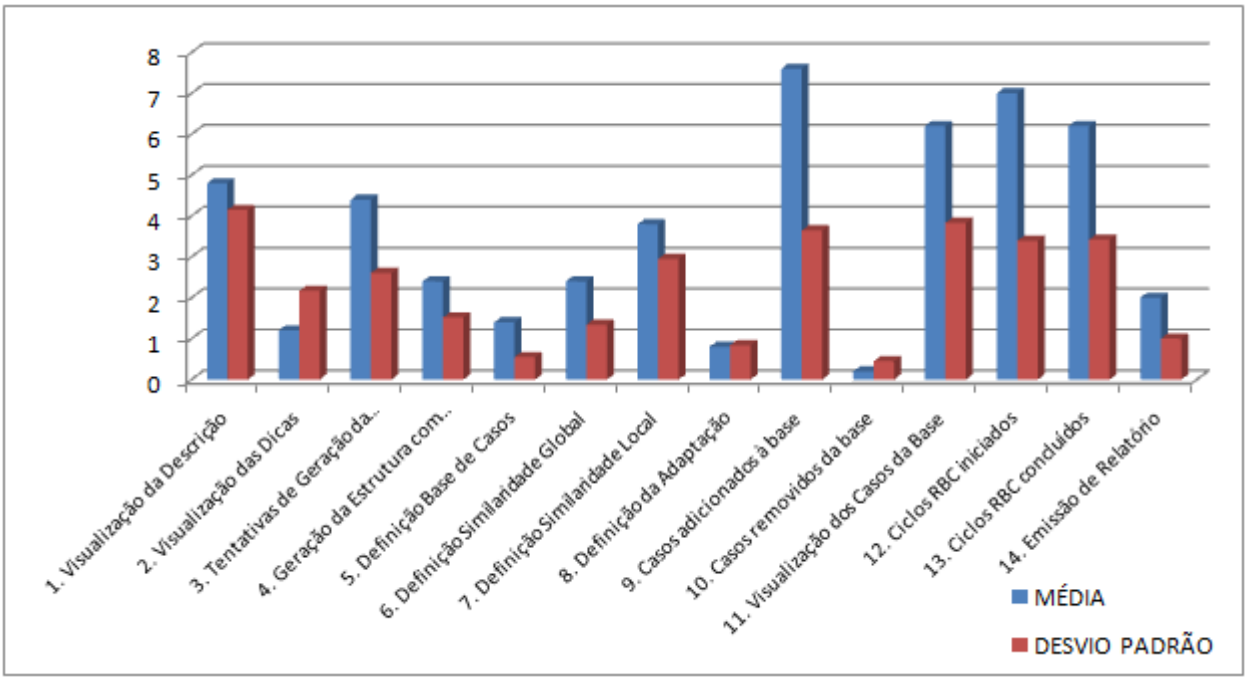


Figura 47. Gráfico das estatísticas do relatório.

O gráfico das estatísticas mostra o número de vezes que cada aluno realizou as 14 funcionalidades ao realizar o exercício. Um quesito que pode ser avaliado é o 3: Número de Tentativas de Geração da Estrutura e o 4: Geração da Estrutura com Sucesso, onde é possível constatar que os alunos em média realizaram 4,4 tentativas e tiveram em média 2,4 vezes de sucesso na geração. Isso indica que alguns alunos tiveram problemas na definição dos atributos. A dificuldade identificada foi pelo problema do nome dos atributos do caso que utilizavam caracteres especiais, o que impossibilitava a execução da aplicação.

Porém, um detalhe observado é que se o aluno iniciou um novo projeto na ferramenta, as estatísticas armazenadas em outros projetos não são mais apresentadas, não demonstrando valores reais, situação está que ocorreu com 4 dos 12 alunos que testaram. Para que isto não ocorra, o professor ao disponibilizar o exercício pode fornecer também o projeto sem nenhuma definição realizada ao aluno, para que ele desenvolva o sistema sempre neste mesmo projeto, fazendo com que as estatísticas sejam reais.

Quanto ao conteúdo disponibilizado nas ajudas e no tutorial, que corresponde à parte da fundamentação teórica desta dissertação, um especialista da Área de Inteligência Artificial (professor da disciplina onde foi testada a ferramenta) realizou a avaliação do material. Esta avaliação observou se o conteúdo estava adequado para alunos e profissionais poderem

compreender o funcionamento dos sistemas de RBC e se poderia ser utilizado com um guia de referência para consultas, em caso de dúvidas durante a concepção e execução de um sistema RBC.

Na avaliação o especialista observou que os conteúdos do tutorial, que corresponde à parte conceitual (teórica) dos sistemas de RBC não estavam relacionados com as formas de execução dentro da ferramenta, onde foi então criada uma ligação dos conceitos do tutorial com as funcionalidades da ferramenta, indicando como cada conceito dos sistemas RBC pode ser explorado dentro do sistema.

Esta ligação foi realizada através de *links* onde se pode realizar a navegação entre os conteúdos do tutorial e das ajudas do sistema. Após este ajuste o especialista validou os conteúdos disponibilizados, indicando que pode ser utilizado como referência para consultas sobre sistemas de RBC.

Outra avaliação realizada foi para verificar se os resultados das aplicações desenvolvidas estavam coerentes, quanto à definição dos parâmetros e métricas de similaridade. Desta forma foi desenvolvido um sistema de RBC proposto por Wangenheim e Wangenheim (2003), um exemplo de aplicação de busca inteligente de catálogos para uma agência de viagens *online.* Nesta aplicação os casos foram compostos por oito atributos (destino da viagem, data da partida, duração da viagem, meio de transporte, categoria da acomodação, preço, serviços adicionais e nome da companhia de viagem), sendo os seis primeiros atributos considerados discriminantes.

As métricas de similaridade (local e global) e pesos foram os mesmos apresentados no exemplo. As métricas de similaridade local utilizadas foram: a)Símbolo Ordenado (para categoria da acomodação); b) Matriz de Similaridade (para destino da viagem e meio de transporte); c) Intervalo (para data da partida); e d) função linear (para duração da viagem e preço). Como métrica de similaridade global foi utilizada a métrica do vizinho mais próximo ponderado por pesos. A adaptação não foi utilizada neste teste, pois o exemplo não descrevia a utilização de métodos.

Depois de realizadas as definições foram adicionados manualmente à base de casos seis casos, conforme citados no exemplo, para que fosse possível realizar uma comparação entre a aplicação desenvolvida e o exemplo. Foram realizados testes idênticos aos citados no exemplo, onde se obteve os mesmos valores de similaridade e casos recuperados da base.

Com este mesmo exemplo, após a comparação com o exemplo, o especialista em IA realizou alterações nas métricas de similaridade e também aplicou métodos de adaptação ao preço

do pacote, onde foi possível verificar o funcionamento do sistema. Os resultados obtidos com os testes foram adequados e coerentes, de acordo com as métricas de similaridade e métodos de adaptação utilizados.

Para concluir, podem-se verificar as hipóteses levantadas no início do trabalho: a) A ferramenta desenvolvida facilita o a criação de sistemas de RBC.; e b) A ferramenta desenvolvida auxilia no ensino da técnica de RBC.

A primeira hipótese foi comprovada através de testes realizados com alunos e especialistas. Foi verificado o tempo de concepção das aplicações desenvolvidas utilizando a ferramenta, sendo considerado bastante reduzido em relação a outras soluções similares. Uma das facilidades constatada pelos usuários foi a disponibilidade de uma interface gráfica simples e objetiva com ajudas contextualizadas. Outra consideração foi a quantidade de métricas de similaridade disponível, considerada satisfatória pelo especialista, pois pode-se aplicar à vários tipos de dados. Também pode-se destacar a facilidade em utilizar fonte de dados existentes, com o mapeamento das estruturas ou ainda a criação de bancos de dados através da própria ferramenta.

A segunda hipótese pode ser confirmada diretamente com o especialista da área, que é professor da Disciplina de Inteligência Artificial, que indicou a ferramenta como sendo de grande relevância no ensino da técnica de RBC, pois apresenta aos alunos cada etapa da concepção de um sistema RBC, além de mostrar a execução de cada etapa do ciclo RBC. Outro ponto que se pode destacar é o quesito 14 do questionário de avaliação de funcionalidade (Na sua opinião a ferramenta pode auxiliar no ensino da técnica de RBC), que foi avaliado de forma unânime como Extremamente, além das observações apresentadas pelos alunos, que conseguiram compreender o funcionamento do ciclo RBC.

Desta forma a ferramenta torna-se disponível para professores e desenvolvedores de sistemas que desejam aplicar a técnica de RBC.

5 CONCLUSÕES

Os avanços na área de Inteligência Artificial são significativos, onde métodos e técnicas buscam simular computacionalmente a capacidade de raciocinar dos seres humanos. A técnica de Raciocínio Baseada em Casos é uma destas técnicas, que pode ser aplicada em várias áreas de conhecimento, pois visa encontrar uma solução para o problema atual comparando-o com casos anteriormente armazenados. O RBC constitui um paradigma de aprendizado de máquina, pois é capaz de simular o raciocínio de um especialista, utilizando a experiência de casos passados na análise e resolução de uma situação atual.

A proposta deste trabalho foi facilitar o desenvolvimento deste tipo de sistema, disponibilizando a alunos de cursos de graduação e pós-graduação, a possibilidade compreender o funcionamento da técnica e as definições envolvidas através da elaboração interativa de sistemas de RBC. Além de aplicações acadêmicas, também é possível realizar as definições na ferramenta e exportá-las para serem utilizado em outras aplicações ou gerar interfaces personalizadas.

Para desenvolver a ferramenta foi inicialmente realizada a revisão bibliográfica sobre RBC, apresentando os conceitos e definições envolvidas com a técnica. As formas de representação de conhecimento usualmente utilizadas e as principais métricas de similaridade, locais e globais, foram analisadas e detalhadas, alcançando os objetivos específicos de pesquisar e analisar a técnica de RBC e analisar as formas de representação do conhecimento e métricas de similaridade.

A pesquisa por soluções similares foi realizada através de livros, artigos e internet, onde foi possível apontar as principais ferramentas para desenvolvimento de sistemas de RBC existentes. Com estas soluções similares foi realizada uma análise das suas principais funcionalidades, apontando as deficiências encontradas e possibilitando definir as características para a ferramenta proposta, alcançando desta forma o objetivo específico de analisar soluções similares.

Na análise, o *framework* jColibri 2 mostrou-se como uma solução viável para ser utilizada como base para a ferramenta proposta, pois conforme descrito pelos desenvolvedores, não possuía uma interface gráfica (ferramenta de autoria) para definição das aplicações desenvolvidas e o conjunto de métricas de similaridade implementadas era pequeno.

Para o objetivo específico analisar uma estratégia de ensino para aplicação em sistemas computacionais foi realizada a pesquisa por abordagens pedagógica, identificando a técnica de

Aprendizagem Baseada em Problemas como apropriada para a proposta do trabalho entre as diferentes estratégias pedagógicas existentes, sendo esta aplicada e verificada sua viabilidade, alcançando o objetivo proposto de aplicá-la em sistemas computacionais.

Para a modelagem foi utilizada a linguagem UML, integrando com o *framework* jColibri 2, que mostrou-se uma biblioteca flexível para o desenvolvimento de sistemas de RBC. Seguindo esta modelagem foi possível desenvolver a ferramenta utilizando a linguagem de programação Java.

A validação foi realizada em uma turma de Inteligência Artificial de um Curso de Ciência da Computação, onde um grupo de 12 alunos desenvolveu sistemas de RBC. Na validação a ferramenta foi disponibilizada aos alunos, que sem a supervisão do mestrando, buscaram criar os sistemas de RBC que foram propostos. A validação foi concluída com uma avaliação através de um formulário de usabilidade e funcionalidade, em conjunto com um especialista em IA, que buscou identificar as principais dificuldades encontradas pelos usuários com o uso da ferramenta.

Buscando melhorar a ferramenta foi realizada a manutenção corretiva e evolutiva, que agora pode ser utilizada por professores no ensino da técnica de RBC. Na ferramenta foram disponibilizadas várias métricas de similaridade globais e locais que podem ser exploradas pelos usuários para cada tipo de aplicação em diferentes domínios de conhecimento.

De forma geral o objetivo do trabalho que foi desenvolver uma ferramenta para construção de Sistemas de Raciocínio Baseado em Casos, com foco no ensino da técnica, para o desenvolvimento de aplicações mostrou-se alcançado e a seguir mostra-se pontualmente as contribuições do trabalho.

5.1 CONTRIBUIÇÕES:

As principais contribuições deste trabalho foram:

- Ferramenta para desenvolvimento de sistemas RBC com foco no ensino da técnica.
- Ferramenta para desenvolvimento ágil de aplicações de RBC;
- Estudo comparativo entre as soluções para desenvolvimento de RBC;
- Implementação e possibilidade de testes de diferentes métricas de similaridade para sistemas de RBC.

5.2 TRABALHOS FUTUROS

Ao longo do desenvolvimento deste trabalho, puderam ser identificadas algumas possibilidades de melhoria e de continuação a partir de futuras pesquisas, as quais incluem:

- Criação de um tutor inteligente: Como o foco é o ensino da técnica de RBC, pode-se criar um sistema tutor inteligente para acompanhar o aluno durante o uso da ferramenta, auxiliando-o no ensino e avaliando o desempenho.

- Possibilidade de criar sistemas híbridos: a ferramenta possibilita o desenvolvimento de sistemas utilizando apenas a técnica de RBC, porém uma possível melhoria seria a utilização técnicas como Algoritmos Genéticos ou Redes Neurais atuando em conjunto com a técnica de RBC, seja na recuperação das informações ou no cálculo da similaridade.

- Possibilitar o uso de ontologias: ontologia é um instrumento de auxílio à recuperação de informações e são extremamente relevantes para a criação de mecanismos de busca mais eficientes. O *framework* jColibri 2 já possibilita o uso de ontologias, porém necessita o uso de aplicações externas para definir a estrutura de dados de armazenamento das informações. O desenvolvimento de uma funcionalidade para modelar ontologias para sistemas de RBC dentro da ferramenta possibilitaria o uso desta técnica.

- Adição de novas métricas de similaridade: foram disponibilizadas cinco métricas de similaridade global e oito métricas de similaridade local. Uma possível melhoria seria a adição de novas métricas .

- Adição de novos métodos de adaptação: foram disponibilizados dois métodos de adaptação, possibilitando adicionar novos métodos para serem aplicados no desenvolvimento de sistemas RBC.

- Geração de interface gráfica: ao exportar uma aplicação desenvolvida é possível executá-la, porém utiliza-se o modo console do Java. Um trabalho futuro poderia ser a criação automática de uma interface gráfica para execução do sistema.

REFERÊNCIAS BIBLIOGRÁFICAS

AAMODT, A.; PLAZA, E. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *In:* **AICOM – Artificial Intelligence Communications**. Vol. 7. Ed. IOS Press, EUA, 1994.

ABEL, M. **Um estudo sobre raciocínio Baseado em Casos**. Porto Alegre. Trabalho Individual (Pós-Graduação em Ciência da Computação) – Universidade Federal do Rio Grande do Sul, 1996.

ABERGO. A certificação do ergonomista brasileiro. *In* **Editorial do Boletim** 1/2000, Associação Brasileira de Ergonomia, 2000.

ALLY, M. Foundations of educational theory for online learning. *In:* ANDERSON, T.; ELLOUMI, F. **Theory and practice of online learning.** São Paulo: Athabasca University, 2004.

ANDRADE, M. A. B. S.; CAMPOS, L. M. L. Aprendizagem Baseada em Problemas no Ensino Médio: O Professor como Tutor. *In:*Encontro Nacional de Pesquisa em Educação em Ciências (ENPEC), 2007, Florianópolis, **Anais ...** 2007.

ART Enterprise. **ART Enterprise**, 2008. Disponível em <http://www.mindbox.com/Products/ARTEnterprise.aspx>. Acesso em 01/07/2008.

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Modern Information Retrival.** ACM Press: New York, 1999.

BASTIEN, C.; SCAPIN, D. **Ergonomic criteria for the evaluation of human-computer interfaces**. Technical Report 156. INRIA - Institut National de Recherche en Informatique et en Automatique, Rocquencourt, France, 1993. Disponível em: <http://www.webmaestro.gouv.qc.ca/ress/Webeduc/ 2000nov/criteres.pdf> Acesso em: 15 set. 2008.

CBR Shell. **CBR Tools from AIAI** . 2008. Disponível em <http://www.aiai.ed.ac.uk/project/cbr/cbrtools.html> . Acesso em 10/11/2008.

CBR*Tools. **CBR*Tools**, 2008. Disponível em <http://www-sop.inria.fr/axis/cbrtools/manual/>. Acesso em 01/07/2008.

CBR Works. **CBR Works**. 2008. Disponível em <http://www.tecinno.com/>. Acesso em 10/04/2008.

COLL, C.; GOTZENS, C.; MONEREO, C.; ONRUBIA, J.; POZO, J.; TAPIA, A. **Psicologia da Aprendizagem no Ensino Médio**. Porto Alegre: Artmed, 2003.

CYBIS, W. A.; PIMENTA, M. S. ; SILVEIRA, T. T. M. C. ; GAMEZ, L. Uma Abordagem Ergonômica para o Desenvolvimento de Sistemas Interativos. *In*: I Simpósio Brasileiro de Interação entre Seres Humanos e Sistemas Computacionais (IHC), 1998, Maringá. **Anais...** 1998.

CYBIS, W.; BETIOL, A.H.;FAUST, R. **Ergonomia e Usabilidade**: Conhecimentos, Métodos e Aplicações. Ed. Novatec, 2007.

DELISLE, R. **Como realizar a aprendizagem baseada em problemas.** Tradução de Vítor Oliveira. Cadernos do Centro de Recursos de Informação e Apoio Pedagógico ASA. Porto: Edições ASA, 2000.

DIAS, C. **Usabilidade na web:** criando portais mais acessíveis. Rio de Janeiro: Alta Books, 2003.

ERGOLIST. **Ergolist**, 2008. Disponível em < http://www.labiutil.inf.ufsc.br/ergolist/index.html>. Acesso em 20 outubro 2008.

ESTEEM. **ESTEEM**, 2008. Disponível em <http://www.stottlerhenke.com/solutions/decision_support/esteem.htm>. Acesso em 01/07/2008.

FAYAD, M.; SCHIMIDT, D.; RALPH, J. **Implementing application frameworks: object-oriented framework at work.** New York: John Wiley & Sons, 1999. 729 p.

FERNANDES, A. M. R. **Inteligência Artificial**: noções gerais. Ed. Visual Books, Florianópolis, 2003.

GAIA - Group of Artificial Intelligence Applications. **JColibri.** 2008. Disponível em <http://gaia.fdi.ucm.es/projects/jcolibri/>. Acesso em 10/11/2008.

GAMMA, E., HELM R., JOHNSON, R. , VLISSIDES, J. **Design Patterns Elements of Reusable Object-Oriented Software.** Ed. Addison-Wesley, 1994.

GLADCHEFF, A.P. **Um instrumento de avaliação da qualidade para software educacional de Matemática.** Dissertação. (Mestrado). USP: Instituto de Matemática e Estatística, São Paulo, 2000.

HERCULIANI, C. E. **Desenvolvimento de um software de autoria para alunos deficientes não-falantes nas atividades de contos e recontos de histórias.** Marília, 2007. Dissertação (Mestrado em Educação) - Programa de Pós-graduação em Educação. Universidade Estadual Paulista Júlio de Mesquita Filho, 2007.

HERRIED, C. F. The Death of Problem-Based Learning? **Journal of College Science Teaching,** Vol. 32 (Mar/Apr), pp. 364-366, 2003.

INDUCE-IT. **Induce-It**, 2008. Disponível em <http://www.inductive.com/softcase.htm>. Acesso em 01/07/2008.

JACZYNSKI, M.; TROUSSE, B. An Object-Oriented Framework for the Design and Implementation of Case-Based Reasoners. In: German Workshop on Case-Based Reasoning, 6th, 1998, Berlin. **Proceedings…** Germany: German Workshop on Case-Based Reasoning, 1998. v. 1. p. 1-10.

JONES, E.A. **Myths About Assessing the Impact of the Problem-Based Learning on Students**. The Journal of General Education. Vol. 51, n° 4, 2002, p. 326-334.

JULIO, M. R. F. M. Um **Estudo de Métricas de Similaridade em Sistemas Baseados em Casos** Aplicados à Área da Saúde. Campinas. Trabalho Final (Mestrado Profissional) - Instituto de Computação, Universidade Estadual de Campinas, 2005.

KOLODNER, J. **Case-based reasoning**. San Mateo: Morgan Kaufmann, 1993.

KOSLOSKY, M.A.N. **Aprendizagem Baseada em Casos, Um Ambiente para Ensino de Lógica de Programação.** Florianópolis. Dissertação (Mestrado em Engenharia de Produção) – Centro Tecnológico, Universidade Federal de Santa Catarina, 1999.

LAGEMANN, G. V. **RBC para o Problema de Suporte ao Cliente nas Empresas de Prestação de Serviço de Software:** O Caso Datasul**.** Dissertação (Mestrado em Engenharia de Produção) – Universidade Federal de Santa Catarina, Florianópolis, 1998.

LEE, R. W. **Pesquisa Jurisprudencial Inteligente.** Florianópolis. Tese (Doutorado em Engenharia de Produção) – Centro Tecnológico, Universidade Federal de Santa Catarina. , 1998

LIMA, D. R., ROSATELLI, M. C. Uma abordagem para a recuperação e adaptação de casos em um Sistema Tutor Inteligente. *In*: Simpósio Brasileiro de Informática na Educação – Universidade Federal do Amazonas, 2004, Amazonas. **Anais ....** Amazonas, 2004.

LUGER, G. F. **Inteligência Artificial**: estruturas e estratégias para a resolução de problemas complexos. 4ª Edição. Ed. Bookmann, Porto Alegre, 2004.

MALANGE, F. C. V., **Uma Proposta De Aprendizado Da Lógica Utilizando Um Ambiente Virtual De Aprendizagem Colaborativ**a. Florianópolis, 2004. Dissertação (Mestrado em Ciência da Computação) - Programa de Pós-Graduação em Ciência da Computação. UFSC, 2005.

MARTINS, J. G. **Aprendizagem Baseada em Problemas Aplicada a Ambiente Virtual de Aprendizagem**. Tese (Doutorado em Engenharia da Produção) - Programa de Pós-Graduação em Engenharia de Produção. Universidade Federal de Santa Catarina. Florianópolis, 2002.

MERGEL, B. **Instructional Design & Learning Theory.** Monografia (Especialização) - Educational Communications and Technology, University of Saskatchewan, Saskatoon, 1998. Disponível em: <http://www.usask.ca/education/coursework/802papers/mergel/mergel.PDF>. Acesso em: 01 out. 2009.

MIRANDA, D. F. **Geometria Táxi, uma métrica para os espaços geográficos e urbanos uma análise exploratória.** 1999. Dissertação (Mestrado em Tratamento de Informação Espacial) - Pontifícia Universidade Católica de Minas Gerais, Belo Horizonte, 1999.

MYCBR. **myCBR.** 2008. Disponível em < http://mycbr-project.net/>. Acesso em 10/11/2008.

NIELSEN, J. **Usability Engineering**. Ed. CA:Academic Press, Boston, EUA, 1992. Disponível em:<http://books.google.com/books?id=o1IqPH0a2fYC&pg=PA264&lpg=PA264&dq=%22Usability+engineering%22+nielsen&sig= Caknr1IEIxXJDnYlDwrh0bsnwvQ&hl=pt-BR> Acesso em: 25 set. 2008.

OLIVEIRA, J. M. P. ; GALANTE, D. ; FERNANDES, C. T. ; SA, E. J. V. ; TEIXEIRA, J. S. F. . Representação da Interação do Aprendiz em Sistemas Hipermídia Adaptativos Educacionais que Utilizam a Aprendizagem Baseada em Problemas como Modelo Pedagógico. *In*: Worshop Sobre Informática na Escola (WIE)/Congresso da Sociedade Brasileira de Computação, 2005, São Leopoldo, RS. **Anais ...**, 2005.

RECIO-GARCIA, J. A.; DIAZ-AGUDO, B.; GONZÁLES-CALERO, R. jCOLIBRI2 Tutorial. **Group for artificial intelligente applications**, Universidad Complutense de Madrid, Set. 2008.

RUIZ-GRANADOS, A. A. S. **jCOLIBRI: estado actual y posibles mejoras:** Aprendizaje Automático. Universidad Complutense de Madrid, Espanha, 2005.

SANTOS, F. J. J. **Sistema De Gerenciamento de Redes Baseado em Conhecimento.** Lavras, 2004. 124f. Monografia de Especialização (Pós-Graduação Lato Sensu Administração em Redes Linux) – Departamento de Ciência da Computação. Universidade Federal de Lavras, Lavras, 2004.

SARDO, P.M.G. **Aprendizagem baseada em problemas em reanimação cárdio-pulmonar no ambiente virtual de aprendizagem Moodle**. Florianópolis, 2007. 228f. Dissertação (Mestrado em Enfermagem) - Programa de Pós Graduação em Enfermagem. Universidade Federal de Santa Catarina. Florianópolis, 2007.

SCHNEIDER, H. N. **Um Ambiente Ergonômico de Ensino-Aprendizagem Informatizado.** Florianópolis, 2002. 162p. Tese (Doutorado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção. Universidade Federal de Santa Catarina. Florianópolis, 2002.

SILVA, F. J. J. **Sistema De Gerenciamento de Redes Baseado em Conhecimento**. Lavras, 2004. 124f. Monografia de Especialização (Pós-Graduação Lato Sensu Administração em Redes Linux) – Departamento de Ciência da Computação. Universidade Federal de Lavras, Lavras, 2004.

SOVAT, R., ALUÍSIO, S.M., CARVALHO, A.C.P.L.F**. "RaBeCa : A Hybrid Case-Based Reasoning Development Environment"** ictai,pp.61, 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01), 2001

THÉ, M.A.L. **Raciocínio Baseado em Casos: Uma Abordagem Fuzzy para Diagnóstico Nutricional**. Tese de Doutorado (Programa de Pós Graduação em Engenharia de Produção). Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, 2001.

VALLE FILHO, A. M. **Um Modelo para Implementação de Consciência em Robôs Móveis.** Florianópolis. Tese (Doutorado em Engenharia de Produção) – Centro Tecnológico, Universidade Federal de Santa Catarina, 2003.

VIDAL, M.C.; **Introdução à Ergonomia**. Fundação Coppetec, 1993.

WALSH, Allin. **The tutor in problem based learning:** A novice's guide. Hamilton: McMaster University, 2005.

WANGENHEIM, C. G.; WANGENHEIM, A. **Raciocínio baseado em casos**. Ed. Manole, Barueri, 2003.

WATSON, I. D. **Applying case-based reasoning.** Ed. Morgan Kaufmann Publishers, USA, 1997.

WOODS, D. R. **Problem-Based Learning**: How to get the most from PBL. MacMaster: McMaster University, 1994.

APÊNDICE A – RELATÓRIO DA APLICAÇÃO

PROJETO_SOJA

PROPRIEDADES DO PROJETO

Nome do Projeto:	PRJSOJA
Data Criação:	Thu Nov 12 18:21:12 BRST 2009
Pasta:	C:\Users\Helton\PRJSOJA.xml
Autor:	Nome do Aluno
Uso:	Realizar exercício
Testado:	True

EXERCÍCIO RELACIONADO

Título:	Exercício de Identificação de Pragas no Cultivo da Soja
Arquivo:	C:\Exercício de Identificação de Pragas no Cultivo da Soja.xml
Descrição:	A tecnologia de Manejo Integrado de Pragas da Soja (MIP-Soja) foi implantada no Brasil, na década de 1970, e tem sido aperfeiçoada constantemente. Essa tecnologia orienta na tomada de decisões de controle de pragas com base num conjunto de informações sobre os insetos e sua densidade populacional, na ocorrência de inimigos naturais e na capacidade da cultura de tolerar os danos. Assim, o monitoramento da lavoura, a identificação correta das pragas e dos inimigos naturais, o conhecimento do estádio de desenvolvimento da planta e dos níveis de ação são importantes componentes do Cultivo da Soja. Portanto este projeto visa facilitar a identificação das pragas que ocorrem no cultivo de soja no Brasil.

FONTE DE DADOS

Tipo:	Banco de Dados
Driver:	HSQL
Usuário:	sa
Servidor:	jdbc:hsqldb:file:C:\Users\Helton\Desktop\Testes Ferramenta\RBC_Soja\RBC_Soja\RBC_PRJ\appRBC\BD\PRJSOJA

DEFINIÇÃO DA BASE DE CASOS

Recuperação:	Base de Casos Linear

ESTRUTURA DO CASO

>>> Descrição

RBC | idCaso ([PK] Integer(0))

RBC | Parenquimas_cortadas (Boolean(0))

RBC | Nervuras_cortadas (Boolean(0))

RBC | Peciolos_cortados (Boolean(0))

RBC | Graos_atacados (Boolean(0))

RBC | Retencao_foliar (Boolean(0))

RBC | Vagem_atacada (Boolean(0))

RBC | Caule_cortado (Boolean(0))

RBC | Pouca_raiz (Boolean(0))

RBC | Ovos_na_folha (Boolean(0))

RBC | Ovos_na_vagem (Boolean(0))

>>> Resultado

RBC | Nome_da_Praga (String(100))

SIMILARIDADE GLOBAL

Métrica:	rbc.similaridade.global.SimGlobalVizMaisProximoPonderado@1fa1bb6
Definição dos Pesos:	Peso fixo (Definido na aplicação)
Número de casos(k):	10
Limiar Similariadde:	0%

SIMILARIDADE LOCAL

RBC | Parenquimas_cortadas	Igualdade (Negação: false | CaseSensitive: false)
RBC | Nervuras_cortadas	Igualdade (Negação: false | CaseSensitive: false)
RBC | Peciolos_cortados	Igualdade (Negação: false | CaseSensitive: false)
RBC | Graos_atacados	Igualdade (Negação: false | CaseSensitive: false)
RBC | Retencao_foliar	Igualdade (Negação: false | CaseSensitive: false)
RBC | Vagem_atacada	Igualdade (Negação: false | CaseSensitive: false)
RBC | Caule_cortado	Igualdade (Negação: false | CaseSensitive: false)
RBC | Pouca_raiz	Igualdade (Negação: false | CaseSensitive: false)
RBC | Ovos_na_folha	Igualdade (Negação: false | CaseSensitive: false)
RBC | Ovos_na_vagem	Igualdade (Negação: false | CaseSensitive: false)

ADAPTAÇÃO

Sem adaptação definida.

BASE DE CASOS

Número Caso	idCaso	parenquimas_cortadas	nervuras_cortadas	peciolos_cortados	graos_atacados	retencao_foliar	vagem_atacada	caule_cortado	pouca_raiz	ovos_na_folha	ovos_na_vagem	idCaso	nome_da_Praga
1	1	true	false	false	true	false	false	false	false	false	false	1	Broca
2	3	true	false	false	false	false	false	false	true	true	false	1	Piolho

ESTATÍSTICAS DA REALIZAÇÃO DO EXERCÍCIO

Quantidade de vezes que foram realizadas cada etapa do exercício

Visualização da Descrição do Problema	9
Visualização das Dicas	5
Tentativas de Geração da Estrutura	4
Geração da Estrutura com Sucesso	4
Definição Base de Casos	1
Definição Similaridade Global	1
Definição Similaridade Local	1
Definição da Adaptação	0
Casos adicionados à base	13
Casos removidos da base	0
Visualização dos Casos da Base	16
Ciclos RBC iniciados	5
Ciclos RBC concluídos	6
Emissão de Relatório	5

APÊNDICE B – ESTRUTURA DAS AJUDAS

Ajuda do Sistema

- Projeto
  - Definir Projeto
  - Visualizar Informações do Projeto
  - Criar Exercício
- Ensino
  - Realizar Exercício
    - Iniciar Exercício
    - Estrutura
    - Definições
    - Execução
  - Aprendizagem Baseada em Problemas
- Estrutura
  - Fonte de Dados
  - Capturar/Definir Estrutura
    - Editar Campo
    - Definir Chave Estrangeira
  - Estrutura do Caso
  - Base de Casos
  - Similaridade Global
  - Similaridade Local
    - Igualdade
    - Intervalo
    - Degrau(Threshold)
    - Função (Linear, Quadrática, Normal)
    - Matriz de Similaridade
    - Categorias ordenadas
    - Maior Sequência de Caracteres
    - Textual
  - Adaptação
- Execução
  - Executar
    - Definir Caso Atual
    - Definir Valor do Atributo
  - Base de Casos
    - Visualizar Informações do Caso
  - Exportar

APÊNDICE C – CÓDIGO FONTE GERADO

Ao criar um projeto na ferramenta e desenvolver o sistema de RBC são gerados códigos e arquivos de configuração que são executados. Ao exportar a aplicação eles podem ser visualizados e utilizados pelos usuários para adequar a novas aplicações fora da ferramenta.

A ferramenta gera as classes em código Java do projeto principal, as classes que armazenam as informações do caso (uma para cada componente: descrição, solução e resultado) e também gera os arquivos de configuração utilizados pelo jColibri.

CLASSE PRINCIPAL DO PROJETO

```
package appRBC;
import jcolibri.connector.PlainTextConnector;
import java.util.ArrayList;
import java.util.List;
import java.util.Collection;
import java.util.HashMap;
import jcolibri.casebase.*;
import java.util.Scanner;
import java.util.Date;
import jcolibri.cbraplications.StandardCBRApplication;
import jcolibri.cbrcore.Attribute;
import jcolibri.cbrcore.CBRCase;
import jcolibri.cbrcore.CBRCaseBase;
import jcolibri.cbrcore.CBRQuery;
import jcolibri.cbrcore.Connector;
import jcolibri.connector.DataBaseConnector;
import jcolibri.exception.ExecutionException;
import jcolibri.method.retrieve.RetrievalResult;
import rbc.similaridade.*;
import rbc.similaridade.local.*;
import rbc.similaridade.global.*;
import jcolibri.method.retrieve.NNretrieval.NNConfig;
import jcolibri.method.retrieve.NNretrieval.NNScoringMethod;
import jcolibri.method.retrieve.selection.SelectCases;
import jcolibri.method.reuse.CombineQueryAndCasesMethod;
import jcolibri.method.reuse.NumericDirectProportionMethod;
import jcolibri.method.retrieve.FilterBasedRetrieval.*;
import jcolibri.util.AttributeUtils;
import jcolibri.cbrcore.CaseComponent;
import jcolibri.method.retrieve.FilterBasedRetrieval.predicates.*;

public class PRJSOJA implements StandardCBRApplication {

    Connector _connector;
    CBRCaseBase _caseBase;
    FilterConfig filterConfig = null;

        public void configure() throws ExecutionException{
            try{
            _connector = new DataBaseConnector();
            _connector.initFromXMLfile(jcolibri.util.FileIO.findFile("appRBC/databaseconfig.xml"));
            _caseBase  = new LinealCaseBase();

        } catch (Exception e){
            throw new ExecutionException(e);
        }
    }

    public CBRCaseBase preCycle() throws ExecutionException {
        _caseBase.init(_connector);
        return _caseBase;
    }

    public void cycle (CBRQuery query) throws ExecutionException {
         try {
                ArrayList<RetrievalResult>                    lista                    =                    new
ArrayList<RetrievalResult>(this.getCasosSelecionados(query,null));
                int i = 0;
                System.out.println("");
                System.out.println("------------------------------------------------------------");
                System.out.println("Casos recuperados (Mais similares)");
                for (RetrievalResult ret: lista) {
                        System.out.println("[ "+i+" ] "+ret);
                   i++;
                }
```

```
                ArrayList<CBRCase> casos = new ArrayList<CBRCase>();
                if (lista.size()==0) {
                        System.out.println("Nenhum caso encontrado");
                        CBRCase cbrCase = new CBRCase();
                        cbrCase.setDescription(query.getDescription());
                        casos.add(cbrCase);
                } else {
                        System.out.println("------------------------------------------------------------");
                        System.out.println("Informe o caso mais similar: ");
                        Scanner sc = new Scanner(System.in);
                        i = sc.nextInt();
                        RetrievalResult ret = lista.get(i);
                        casos.add(ret.get_case());
                }
                if (casos.size()==1) {
                        Collection<CBRCase> casosAdapt = this.getCasosAdaptados(query, casos);
                        CBRCase caso = casosAdapt.iterator().next();
                        System.out.println("------------------------------------------------------------");
                        System.out.println("Caso adaptado:");
                        System.out.println(caso);
                        System.out.println("------------------------------------------------------------");
                        System.out.println("Revisão:");
                        if (caso.getDescription()!=null) this.getDadosCasoAtual(caso.getDescription(), true);
                        if (caso.getSolution()!=null) this.getDadosCasoAtual(caso.getSolution(), true);
                        if (caso.getResult()!=null) this.getDadosCasoAtual(caso.getResult(), true);
                        System.out.println("------------------------------------------------------------");
                        System.out.println("Retenção:");
                        this.setNovoIdCaso(caso);
                        this.gravaCaso(caso);
                        System.out.println("------------------------------------------------------------");
                        System.out.println("Caso retido com sucesso!");
                        System.out.println("Fim do ciclo RBC");
                }
        } catch (Exception erro) {
                erro.printStackTrace();
        }
    }

    public void postCycle() throws ExecutionException {
        this._caseBase.close();
        if (this._connector!=null) this._connector.close();
    }

    public List<RetrievalResult> getTodosCasos () throws Exception {
        Collection<CBRCase> casos = FilterBasedRetrievalMethod.filterCases(_caseBase.getCases(), null, null);
        ArrayList<RetrievalResult> lista = new ArrayList<RetrievalResult>();
        for (CBRCase caso : casos) {
            RetrievalResult ret = new RetrievalResult(caso,0.0);
            lista.add(ret);
        }
        return lista;
    }

    public  Collection<RetrievalResult>  getCasosSelecionados  (CBRQuery  query,  List<Integer>  pesos)  throws
ExecutionException {
        NNConfig simConfig = new NNConfig();
        simConfig.setDescriptionSimFunction(new SimGlobalVizMaisProximoPonderado());

        Attribute _Parenquimas_cortadas = new Attribute("parenquimas_cortadas", RBCDescricao.class);
        SimLocalIgualdade _sim_Parenquimas_cortadas = new SimLocalIgualdade();
        _sim_Parenquimas_cortadas.setNegacao(false);
        _sim_Parenquimas_cortadas.setCaseSensitive(false);
        simConfig.addMapping(_Parenquimas_cortadas, _sim_Parenquimas_cortadas);
        simConfig.setWeight(_Parenquimas_cortadas, 1.0);

        Attribute _Nervuras_cortadas = new Attribute("nervuras_cortadas", RBCDescricao.class);
        SimLocalIgualdade _sim_Nervuras_cortadas = new SimLocalIgualdade();
        _sim_Nervuras_cortadas.setNegacao(false);
        _sim_Nervuras_cortadas.setCaseSensitive(false);
        simConfig.addMapping(_Nervuras_cortadas, _sim_Nervuras_cortadas);
        simConfig.setWeight(_Nervuras_cortadas, 1.0);

        Attribute _Peciolos_cortados = new Attribute("peciolos_cortados", RBCDescricao.class);
        SimLocalIgualdade _sim_Peciolos_cortados = new SimLocalIgualdade();
        _sim_Peciolos_cortados.setNegacao(false);
        _sim_Peciolos_cortados.setCaseSensitive(false);
        simConfig.addMapping(_Peciolos_cortados, _sim_Peciolos_cortados);
        simConfig.setWeight(_Peciolos_cortados, 1.0);

        Attribute _Graos_atacados = new Attribute("graos_atacados", RBCDescricao.class);
        SimLocalIgualdade _sim_Graos_atacados = new SimLocalIgualdade();
        _sim_Graos_atacados.setNegacao(false);
        _sim_Graos_atacados.setCaseSensitive(false);
        simConfig.addMapping(_Graos_atacados, _sim_Graos_atacados);
        simConfig.setWeight(_Graos_atacados, 1.0);

        Attribute _Retencao_foliar = new Attribute("retencao_foliar", RBCDescricao.class);
        SimLocalIgualdade _sim_Retencao_foliar = new SimLocalIgualdade();
        _sim_Retencao_foliar.setNegacao(false);
        _sim_Retencao_foliar.setCaseSensitive(false);
        simConfig.addMapping(_Retencao_foliar, _sim_Retencao_foliar);
        simConfig.setWeight(_Retencao_foliar, 1.0);

        Attribute _Vagem_atacada = new Attribute("vagem_atacada", RBCDescricao.class);
        SimLocalIgualdade _sim_Vagem_atacada = new SimLocalIgualdade();
```

```
        _sim_Vagem_atacada.setNegacao(false);
        _sim_Vagem_atacada.setCaseSensitive(false);
        simConfig.addMapping(_Vagem_atacada, _sim_Vagem_atacada);
        simConfig.setWeight(_Vagem_atacada, 1.0);

        Attribute _Caule_cortado = new Attribute("caule_cortado", RBCDescricao.class);
        SimLocalIgualdade _sim_Caule_cortado = new SimLocalIgualdade();
        _sim_Caule_cortado.setNegacao(false);
        _sim_Caule_cortado.setCaseSensitive(false);
        simConfig.addMapping(_Caule_cortado, _sim_Caule_cortado);
        simConfig.setWeight(_Caule_cortado, 1.0);

        Attribute _Pouca_raiz = new Attribute("pouca_raiz", RBCDescricao.class);
        SimLocalIgualdade _sim_Pouca_raiz = new SimLocalIgualdade();
        _sim_Pouca_raiz.setNegacao(false);
        _sim_Pouca_raiz.setCaseSensitive(false);
        simConfig.addMapping(_Pouca_raiz, _sim_Pouca_raiz);
        simConfig.setWeight(_Pouca_raiz, 1.0);

        Attribute _Ovos_na_folha = new Attribute("ovos_na_folha", RBCDescricao.class);
        SimLocalIgualdade _sim_Ovos_na_folha = new SimLocalIgualdade();
        _sim_Ovos_na_folha.setNegacao(false);
        _sim_Ovos_na_folha.setCaseSensitive(false);
        simConfig.addMapping(_Ovos_na_folha, _sim_Ovos_na_folha);
        simConfig.setWeight(_Ovos_na_folha, 1.0);

        Attribute _Ovos_na_vagem = new Attribute("ovos_na_vagem", RBCDescricao.class);
        SimLocalIgualdade _sim_Ovos_na_vagem = new SimLocalIgualdade();
        _sim_Ovos_na_vagem.setNegacao(false);
        _sim_Ovos_na_vagem.setCaseSensitive(false);
        simConfig.addMapping(_Ovos_na_vagem, _sim_Ovos_na_vagem);
        simConfig.setWeight(_Ovos_na_vagem, 1.0);

        Collection<CBRCase>    casos    =    FilterBasedRetrievalMethod.filterCases(_caseBase.getCases(),    query,
filterConfig);
        Collection<RetrievalResult> eval = NNScoringMethod.evaluateSimilarity(casos, query, simConfig);
        Collection<RetrievalResult> selectedcases = SelectCases.selectTopKRR(eval, 10);
        return selectedcases;
    }

    public  Collection<CBRCase>  getCasosAdaptados  (CBRQuery  query,  Collection<CBRCase>  selectedCases  )  throws
Exception {
        Collection<CBRCase> newcases = selectedCases;
        return newcases;
    }

    public void setNovoIdCaso (CBRCase melhorCaso) throws Exception {
         if (melhorCaso==null) return;
         Object id = melhorCaso.getID();
         if ((id==null) || (melhorCaso.getID() instanceof Integer) || (melhorCaso.getID() instanceof Float)) {
                 Collection<CBRCase> casos = _caseBase.getCases();
         Object maxId = 0;
         if ((casos!=null)&&(casos.size()>0)) {
             ArrayList<CBRCase> lista = new ArrayList<CBRCase>(casos);
                 for (CBRCase caso : lista) {
                      id = caso.getID();
                     if (id == null) id = (Integer) 0; else
                      if ((id instanceof Integer)&&((Integer)id > (Integer) maxId)) maxId = (Integer) id; else
                      if ((id instanceof Float)&&((Float)id > (Float) maxId)) maxId = (Float) id;
                 }
         }
            if (id == null) id = new Integer(1);
         if (id instanceof Integer) id = ((Integer)maxId)+1;
         if (id instanceof Float) id = ((Float)maxId)+1;
         } else
         if (id instanceof String) id = ((String)id)+"_New";
         HashMap<Attribute, Object> componentsKeys = new HashMap<Attribute,Object>();
         if   (melhorCaso.getDescription()!=null)   componentsKeys.put(melhorCaso.getDescription().getIdAttribute(),
id);
         if (melhorCaso.getSolution()!=null) componentsKeys.put(melhorCaso.getSolution().getIdAttribute(), id);
         if (melhorCaso.getResult()!=null) componentsKeys.put(melhorCaso.getResult().getIdAttribute(), id);
         jcolibri.method.revise.DefineNewIdsMethod.defineNewIdsMethod(melhorCaso, componentsKeys);
    }

    public void gravaCaso(CBRCase melhorCaso) throws Exception {
        jcolibri.method.retain.StoreCasesMethod.storeCase(_caseBase, melhorCaso);
    }

    public void removeCaso (CBRCase caso) throws Exception {
        if ((caso == null)||(_caseBase==null)) return;
        ArrayList<CBRCase> lista = new ArrayList<CBRCase>();
        lista.add(caso);
        _connector.deleteCases(lista);
        _caseBase.init(_connector);
    }

    public CBRCase getCasoIgual (CBRCase casoProcura) {
        if ((_caseBase == null)||(_caseBase.getCases()==null)||(_caseBase.getCases().size()==0)) return null;
        for (CBRCase caso: _caseBase.getCases()) {
            boolean diferente = false;
            CaseComponent c = null;
            int i = 0;
            while ((i<3)&&(! diferente)) {
                switch (i) {
                case 0: c = caso.getDescription(); i++; break;
```

```
                case 1: c = caso.getSolution(); i++; break;
                case 2: c = caso.getResult(); i++; break;
                case 3: break;
                }
                if (c == null) continue;
                for (Attribute att: AttributeUtils.getAttributes(c)) {
                    if (! att.equals(c.getIdAttribute())) {;
                        Object valorProcura = AttributeUtils.findValue(att, casoProcura);
                        Object valorCaso = AttributeUtils.findValue(att, caso);
                        if ((valorProcura!=null)&&(valorCaso!=null)&&(! valorProcura.equals(valorCaso))) {
                            diferente = true;
                            break;
                        }
                    }
                }
            }
            if (! diferente) return caso;
        }
        return null;
    }

    @SuppressWarnings("deprecation")
    public void getDadosCasoAtual (CaseComponent comp, boolean revisao) {
        if (! revisao) {
            System.out.println("");
            System.out.println("");
            System.out.println("");
            System.out.println("-------------------------------------------------------------");
            System.out.println("                          RBC Educ                           ");
            System.out.println("-------------------------------------------------------------");
            System.out.println("Informe as informações do caso atual:");
        } else {
            System.out.println("Tecle ? para aceitar o valor original entre <> ou digite o novo valor");
        }
        Collection<Attribute> lista = AttributeUtils.getAttributes(comp);
        Scanner sc = new Scanner(System.in);
        for (Attribute att: lista) {
            if (att.equals(comp.getIdAttribute())) continue;
            if (revisao)
                System.out.println(att.getName()+"<"+AttributeUtils.findValue(att, comp)+">");
            else
                System.out.println(att.getName()+": ");
            String valor = sc.next();
            if ((valor!=null)&&(!valor.equals("?"))) {
                Object valorObj = null;
                if (att.getType().equals(Integer.class))  valorObj = new Integer(valor);
                if (att.getType().equals(String.class))   valorObj = valor;
                if (att.getType().equals(Float.class))    valorObj = new Float(valor);
                if (att.getType().equals(Character.class))valorObj = new Character(valor.charAt(0));
                if (att.getType().equals(Boolean.class))  valorObj = new Boolean(valor);
                if (att.getType().equals(Date.class))     valorObj = new Date(valor);
                AttributeUtils.setValue(att, comp, valorObj);
            }
        }
    }

    public static void main(String[] args) {
        PRJSOJA rbc = new PRJSOJA();
        try {
            rbc.configure();
            rbc.preCycle();
            RBCDescricao queryDesc = new RBCDescricao();
            rbc.getDadosCasoAtual(queryDesc,false);
            CBRQuery query = new CBRQuery();
            query.setDescription(queryDesc);
            rbc.cycle(query);
            rbc.postCycle();

        } catch (Exception e) {
            System.out.println(e.getMessage());
            e.printStackTrace();
        }
    }

}
```

CLASSE DA DESCRIÇÃO DO CASO

```
package appRBC;

import jcolibri.cbrcore.Attribute;
import jcolibri.cbrcore.CaseComponent;

public class RBCDescricao implements CaseComponent {

    private Integer idCaso;
    private Boolean parenquimas_cortadas;
    private Boolean nervuras_cortadas;
    private Boolean peciolos_cortados;
    private Boolean graos_atacados;
    private Boolean retencao_foliar;
    private Boolean vagem_atacada;
    private Boolean caule_cortado;
    private Boolean pouca_raiz;
```

```
    private Boolean ovos_na_folha;
    private Boolean ovos_na_vagem;

    public RBCDescricao() {
    }

    public String toString() {
        return  ""+idCaso+"  |  "+parenquimas_cortadas+"  |  "+nervuras_cortadas+"  |  "+peciolos_cortados+"  |
"+graos_atacados+"  |  "+retencao_foliar+"  |  "+vagem_atacada+"  |  "+caule_cortado+"  |  "+pouca_raiz+"  |
"+ovos_na_folha+" | "+ovos_na_vagem;
    }

    public Attribute getIdAttribute() {
        return new Attribute("idCaso", this.getClass());
    }


    public Integer getIdCaso() {
        return this.idCaso;
    }

    public void setIdCaso(Integer idCaso) {
        this.idCaso = idCaso;
    }

    public Boolean getParenquimas_cortadas() {
        return this.parenquimas_cortadas;
    }

    public void setParenquimas_cortadas(Boolean parenquimas_cortadas) {
        this.parenquimas_cortadas = parenquimas_cortadas;
    }

    public Boolean getNervuras_cortadas() {
        return this.nervuras_cortadas;
    }

    public void setNervuras_cortadas(Boolean nervuras_cortadas) {
        this.nervuras_cortadas = nervuras_cortadas;
    }

    public Boolean getPeciolos_cortados() {
        return this.peciolos_cortados;
    }

    public void setPeciolos_cortados(Boolean peciolos_cortados) {
        this.peciolos_cortados = peciolos_cortados;
    }

    public Boolean getGraos_atacados() {
        return this.graos_atacados;
    }

    public void setGraos_atacados(Boolean graos_atacados) {
        this.graos_atacados = graos_atacados;
    }

    public Boolean getRetencao_foliar() {
        return this.retencao_foliar;
    }

    public void setRetencao_foliar(Boolean retencao_foliar) {
        this.retencao_foliar = retencao_foliar;
    }

    public Boolean getVagem_atacada() {
        return this.vagem_atacada;
    }

    public void setVagem_atacada(Boolean vagem_atacada) {
        this.vagem_atacada = vagem_atacada;
    }

    public Boolean getCaule_cortado() {
        return this.caule_cortado;
    }

    public void setCaule_cortado(Boolean caule_cortado) {
        this.caule_cortado = caule_cortado;
    }

    public Boolean getPouca_raiz() {
        return this.pouca_raiz;
    }

    public void setPouca_raiz(Boolean pouca_raiz) {
        this.pouca_raiz = pouca_raiz;
    }

    public Boolean getOvos_na_folha() {
        return this.ovos_na_folha;
    }

    public void setOvos_na_folha(Boolean ovos_na_folha) {
        this.ovos_na_folha = ovos_na_folha;
```

```
    }

    public Boolean getOvos_na_vagem() {
        return this.ovos_na_vagem;
    }

    public void setOvos_na_vagem(Boolean ovos_na_vagem) {
        this.ovos_na_vagem = ovos_na_vagem;
    }
    public String[] get_Struct() {
        return new String[] { "idCaso" , "parenquimas_cortadas" , "nervuras_cortadas" , "peciolos_cortados" ,
"graos_atacados" , "retencao_foliar" , "vagem_atacada" , "caule_cortado" , "pouca_raiz" , "ovos_na_folha" ,
"ovos_na_vagem" };
    }

    public String[] get_Descricao() {
        return new String[] { "idCaso" , "Parenquimas_cortadas" , "Nervuras_cortadas" , "Peciolos_cortados" ,
"Graos_atacados" , "Retencao_foliar" , "Vagem_atacada" , "Caule_cortado" , "Pouca_raiz" , "Ovos_na_folha" ,
"Ovos_na_vagem" };
    }

    public String[] get_Values() {
        return new String[] { (this.idCaso==null)?"":idCaso.toString()
        , (this.parenquimas_cortadas==null)?"":parenquimas_cortadas.toString()
        , (this.nervuras_cortadas==null)?"":nervuras_cortadas.toString()
        , (this.peciolos_cortados==null)?"":peciolos_cortados.toString()
        , (this.graos_atacados==null)?"":graos_atacados.toString()
        , (this.retencao_foliar==null)?"":retencao_foliar.toString()
        , (this.vagem_atacada==null)?"":vagem_atacada.toString()
        , (this.caule_cortado==null)?"":caule_cortado.toString()
        , (this.pouca_raiz==null)?"":pouca_raiz.toString()
        , (this.ovos_na_folha==null)?"":ovos_na_folha.toString()
        , (this.ovos_na_vagem==null)?"":ovos_na_vagem.toString()
        };
    }
}
```

CLASSE DE RESULTADO DO CASO

```
package appRBC;

import jcolibri.cbrcore.Attribute;
import jcolibri.cbrcore.CaseComponent;

public class RBCResultado implements CaseComponent {

    private Integer idCaso;
    private String nome_da_Praga;

    public RBCResultado() {
    }

    public String toString() {
        return ""+idCaso+" | "+nome_da_Praga;
    }

    public Attribute getIdAttribute() {
        return new Attribute("idCaso", this.getClass());
    }


    public Integer getIdCaso() {
        return this.idCaso;
    }

    public void setIdCaso(Integer idCaso) {
        this.idCaso = idCaso;
    }

    public String getNome_da_Praga() {
        return this.nome_da_Praga;
    }

    public void setNome_da_Praga(String nome_da_Praga) {
        this.nome_da_Praga = nome_da_Praga;
    }
    public String[] get_Struct() {
        return new String[] { "nome_da_Praga" };
    }

    public String[] get_Descricao() {
        return new String[] { "Nome_da_Praga" };
    }

    public String[] get_Values() {
        return new String[] { (this.nome_da_Praga==null)?"":nome_da_Praga.toString()
    };
}
```

CLASSE DE CONFIGURAÇÃO DA APLICAÇÃO

```
<DataBaseConfiguration>
        <HibernateConfigFile>appRBC/hibernate.cfg.xml</HibernateConfigFile>
        <DescriptionMappingFile>appRBC/PRJSOJADescricao.hbm.xml</DescriptionMappingFile>
        <DescriptionClassName>RBCDescricao</DescriptionClassName>
        <ResultMappingFile>appRBC/PRJSOJAResultado.hbm.xml</ResultMappingFile>
        <ResultClassName>RBCResultado</ResultClassName>
</DataBaseConfiguration>
```

CLASSE DE CONFIGURAÇÃO DO BANCO DE DADOS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE     hibernate-configuration     PUBLIC     "-//Hibernate/Hibernate     Configuration     DTD     3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
        <session-factory>
                <property name="connection.driver_class">org.hsqldb.jdbcDriver</property>
                <property name="connection.url">jdbc:hsqldb:file:C:\PRJSOJA</property>
                <property name="connection.username">sa</property>
                <property name="connection.password"></property>
                <property name="connection.pool_size">1</property>
                <property name="dialect">org.hibernate.dialect.HSQLDialect</property>
                <property name="current_session_context_class">thread</property>
                <property name="show_sql">true</property>
        </session-factory>
</hibernate-configuration>
```

CLASSE DE CONFIGURAÇÃO DA DESCRIÇÃO DO CASO

```
<?xml version="1.0"?>
<!DOCTYPE         hibernate-mapping         PUBLIC         "-//Hibernate/Hibernate         Mapping         DTD//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping default-lazy="false">
<class name="appRBC.RBCDescricao" table="RBC">
    <id name="idCaso" column="idCaso" >
    </id>
    <property name="parenquimas_cortadas" column="Parenquimas_cortadas"/>
    <property name="nervuras_cortadas" column="Nervuras_cortadas"/>
    <property name="peciolos_cortados" column="Peciolos_cortados"/>
    <property name="graos_atacados" column="Graos_atacados"/>
    <property name="retencao_foliar" column="Retencao_foliar"/>
    <property name="vagem_atacada" column="Vagem_atacada"/>
    <property name="caule_cortado" column="Caule_cortado"/>
    <property name="pouca_raiz" column="Pouca_raiz"/>
    <property name="ovos_na_folha" column="Ovos_na_folha"/>
    <property name="ovos_na_vagem" column="Ovos_na_vagem"/>
</class>
```

CLASSE DE CONFIGURAÇÃO DO RESULTADO DO CASO

```
<?xml version="1.0"?>
<!DOCTYPE         hibernate-mapping         PUBLIC         "-//Hibernate/Hibernate         Mapping         DTD//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping default-lazy="false">
<class name="appRBC.RBCResultado" table="RBC">
    <id name="idCaso" column="idCaso" >
    </id>
    <property name="nome_da_Praga" column="Nome_da_Praga"/>
</class>
```

APÊNDICE D – FICHA DE AVALIAÇÃO DA FERRAMENTA

AVALIAÇÃO DE SOFTWARE (RBC Educ)

Sobre o Avaliador:

Nome:

Formação (Curso/Fase):

Data Avaliação:

Nível de conhecimento RBC
(Antes da utilização da Ferramenta)

- ☐ Nenhum
- ☐ Básico (Apenas conceitos da técnica)
- ☐ Intermediário (Já desenvolveu aplicações simples)
- ☐ Avançado (Pleno conhecimento da técnica)

Avaliação da Usabilidade da Ferramenta (Baseado nas Heurísticas de Nielsen)

Ajuda aos usuários no reconhecimento, diagnóstico e correção de erros

Quando você faz alguma coisa errada e aparece uma mensagem dizendo que você errou e mostrando o que deve ser feito para arrumar o erro, você consegue entender estas instruções?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

É facil de fazer o que a mensagem sugere?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Controle do Usuário e Liberdade

Você acha que é fácil corrigir um erro, quando por exemplo, você deseja refazer ou desfazer alguma coisa?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Documentação e Ajuda aos Usuários

Ao utilizar o sistema de ajuda (help), foi fácil encontrar a explicação para dúvida?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Foi fácil fazer o que o sistema de ajuda recomendou em relação a uma tarefa que você não havia conseguido realizar?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Reconhecimento aos Invés da Recorrência à Memória do Usuário

Quando você foi executar uma tarefa que já havia realizado antes, foi fácil lembrar como deveria ser feita?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

É fácil compreender a função de cada objeto colocado no software, ou seja, o que cada "coisa" faz?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Relação entre o Sistema e o Mundo Real

Você está achando fácil de ler e entender o que aparece na tela?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

As orientações que o software dá sobre como utilizá-lo são fáceis de serem entendidas?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Quando o software coloca uma definição (conceito) para algo sobre RBC, você consegue entendê-la com facilidade?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Você consegue entender o quê os exercícios pedem para você fazer?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Simplicidade e Estética do Sistema

Você consegue encontrar na tela, com facilidade, onde deve "clicar" para realizar alguma tarefa?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

É fácil entender qual o caminho que deve ser seguido para ir de uma etapa para outra?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Você acha que é fácil perceber quando o software mostra que a execução terminou?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Satisfação Subjetiva

Você gosta de como o software apresenta as informações do projeto?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Você tem vontade de usar outras vezes este software?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Você conseguiu compreender os conceitos e funcionamento da técnicas?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Enfim, você gostou do software?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Sobre o Funcionamento da Ferramenta:

Você conseguiu criar e abrir um projeto na ferramenta?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Você conseguiu definir a fonte de dados e a estrutura do caso?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Você conseguiu definir as métricas de similaridade local e global?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

As métricas de similaridade disponibilizadas na ferramenta são suficientes?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Você conseguiu realizar a adaptação de atributos do caso?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Você conseguiu executar o Ciclo RBC na ferramenta?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Você conseguiu realizar por completo o exercício proposto?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Ao precisar de ajuda, você conseguiu obter as informações que precisava?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

As informações apresentadas no Tutorial RBC são suficientes para o entendimento da técnica RBC?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Na sua opinião a ferramenta pode auxiliar no ensino da técnica de RBC?

☐ Extremamente ☐ Razoavelmente ☐ Ligeiramente ☐ Neutro/Não sei ☐ Nenhum pouco/não

Quais as maiores dificuldades encontradas com o uso da ferramenta?

O que você pode indicar como diferencial da ferramenta em relação a outras soluções?