UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

BRUNO EDUARDO FOLLMANN

BUSCA EM REPOSITÓRIO DE ARTEFATOS DE SOFTWARE UTILIZANDO RACIOCÍNIO BASEADO EM CASOS

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2017

BRUNO EDUARDO FOLLMANN

BUSCA EM REPOSITÓRIO DE ARTEFATOS DE SOFTWARE UTILIZANDO RACIOCÍNIO BASEADO EM CASOS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Engenharia de Computação, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de bacharel.

Orientador: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Engenharia de Computação

TERMO DE APROVAÇÃO

Às 15 horas e 30 minutos do dia 06 de julho de 2017, na sala V006, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, reuniu-se a banca examinadora composta pelos professores Beatriz Terezinha Borsoi (orientador), Kathya Silvia Collazos Linares e Viviane Dal Molin de Souza para avaliar o trabalho de conclusão de curso com o título **Busca em repositório de artefatos de software utilizando raciocínio baseado em casos**, do aluno **Bruno Eduardo Follmann**, matrícula 00656801, do curso de Engenharia de Computação. Após a apresentação o candidato foi arguido pela banca examinadora. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

_____
Beatriz Terezinha Borsoi
Orientador (UTFPR)

_____
Profa. Kathya Silvia Collazos Linares
(UTFPR)

_____
Profa. Viviane Dal Molin de Souza
(UTFPR)

_____
Profa. Beatriz Terezinha Borsoi
Coordenador de TCC

_____
Prof. Pablo Gauterio Cavalcanti
Coordenador do Curso de
Engenharia de Computação

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

FOLLMANN, Bruno Eduardo. Busca em repositório de artefatos de software utilizando raciocínio baseado em casos. 2017. 66 f. Monografia (Trabalho de Conclusão de Curso 2) - Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2017.

O reuso no desenvolvimento de software tem como um dos seus objetivos promover a economia de recursos no desenvolvimento e aumentar a qualidade do software produzido. Artefatos de software são definidos como quaisquer produtos armazenáveis resultantes das atividades pertencentes ao ciclo de vida de software. A economia de recursos se deve à não necessidade de desenvolver artefatos já implementados em outros projetos. A qualidade provém do seu uso em projetos anteriores, garantindo que tenham sido mais amplamente testados. Há dois fatores relevantes em relação às soluções que visam promovam reuso de artefatos: armazenamento e recuperação. O armazenamento refere-se à caracterização que está associada aos metadados dos artefatos. A recuperação relaciona-se às técnicas e procedimentos de busca. Na automatização dos repositórios de artefatos, essas técnicas e procedimentos são implementados e os seus algoritmos para realizá-las são, geralmente, denominados mecanismos de busca. Técnicas caracterizadas como de Inteligência Artificial têm sido empregadas visando promover maior efetividade para tais mecanismos. Visando contribuir para atenuar problemas no desenvolvimento de software por meio do reuso de artefatos, este trabalho apresenta um mecanismo para realizar buscas em um repositório de artefatos de software usando a técnica de Inteligência Artificial denominada Raciocínio Baseado em Casos (RBC). Esse mecanismo que atua em um repositório anteriormente implementado tem por objetivo possibilitar o reuso de artefatos no processo de desenvolvimento de software. A eficiência pretendida com o mecanismo está em recuperar os artefatos mais adequados, entre os armazenados no repositório, de acordo com os critérios indicados para a busca. Esses critérios são definidos por palavras-chave. Palavras-chave que definem o problema de busca caracterizam o projeto. Artefatos associados também são caracterizados por palavras-chave e definem a solução do problema. Assim, um projeto e seus artefatos relacionados constituem um caso no contexto do RBC. A proposta dos metadados para caracterizar o projeto e os artefatos e a organização da estrutura do mecanismo de busca RBC são objetivos deste trabalho. Essa organização define o uso da técnica RBC na implementação de um sistema computacional caracterizado como mecanismo de busca. Esse mecanismo tem como objetivo recuperar o caso mais semelhante aos critérios de busca definidos. Pares de atributo-valor representam esses critérios. Um peso associado a cada par indica a importância da referida palavra-chave na caracterização de cada artefato e como critério de busca.

Palavras-chave: Raciocínio Baseado em Casos. Artefatos de software. Reuso de software.

ABSTRACT

FOLLMANN, Bruno Eduardo. Search in a software artifact repository using case-based reasoning. 2017. 66f. Monografia (Trabalho de Conclusão de Curso 2) - Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2017.

Reuse in software development has as its main goals promoting the saving of resources in the development of projects and increasing their quality. Software artifacts are defined as any of the storable products resulting of the activities pertaining to the software's life cycle. The saving of resources is due to the non-necessity of developing artifacts that were already implemented in other projects. The quality comes of the previous use of the artifacts in other projects, ensuring that they have been more thoroughly tested. There are two relevant factors in relation to the proposal of solutions that promote reuse: the storage and retrieval of artifacts. The storage refers to the characterization that is associated to the artifacts' metadata. The retrieval relates to the search's techniques and procedures. In the automation of artifact repositories, those techniques and procedures are implemented and their algorithms are denominated search engines. Techniques characterized as Artificial Intelligence are being employed in order to promote increased effectiveness for such engines. In order to contribute in the attenuation of problems in the software development by the artifact reuse, this research presents the fundamentals of an engine for performing searches in a software artifacts repository using the Artificial Intelligence technique denominated Case-Based Reasoning (CBR). Such tool has the goal of providing an efficient mean allowing for artifacts reuse in the process of software development. The efficiency is in recovering artifacts that can be used for developing a software project. Keywords that define the problem characterize the project. The associated artifacts are also characterized by keywords and define the solution for the problem. Thus, a project and its associated artifacts constitute a case in the context of CBR. The proposal of the metadata for characterizing the project and the artifacts, as well as the organization of the structure of the search engine, is the goal of this work. That organization defined the use of the CBR technique for the implementation of a computational system characterized as a search engine. That engine has as its objective retrieving the most similar case to the defined search criteria. Attribute-value pairs represent these criteria. A weight associated to each pair indicates the importance of said keyword in the characterization of each artifact and as a search criterion.

Keywords: Case-Based Reasoning. Software artifacts. Software reuse.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS E ABREVIATURAS

IA	Inteligência Artificial
MOF	*Meta Object Facility*
OMG	*Object Management Group*
RBC	Raciocínio Baseado em Casos
RRC	Redes de Recuperação de Casos
SQL	*Structured Query Language*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta o contexto no qual se insere a proposta do trabalho, o problema, os objetivos e a justificativa. O capítulo é finalizado com a apresentação dos capítulos subsequentes que compõem o texto.

1.1 CONSIDERAÇÕES INICIAIS

No desenvolvimento de software, artefato é qualquer resultado obtido da realização de atividades do ciclo de vida de software e que possa de alguma forma ser armazenado. O processo de desenvolvimento de software lida essencialmente com a criação e a utilização de artefatos. Além dos mais evidentes produtos que conformam à classificação de artefato, como os algoritmos na forma de componentes de software, estão os documentos gerados (diagramas de classe, casos de uso, planos de testes e outros), os procedimentos empregados (como os planos, roteiros, métodos e técnicas) e os aspectos de negócio (avaliações de risco e estudos de mercado, por exemplo).

O desenvolvimento de software, seja de caráter econômico ou de cunho científico, é um processo no qual se almeja economia e qualidade. A economia de recursos, de pessoas e de tempo está relacionada ao aumento de produtividade. A qualidade se refere à melhoria do produto que está associada ao atendimento a determinados padrões de qualidade requeridos ao produto final, sejam esses padrões definidos pelo mercado, pelo cliente e usuários, pela própria organização que desenvolve software ou estabelecidos por entidades específicas. A melhoria do produto é obtida por meio da qualificação dos processos de software.

O reuso de artefatos é um procedimento que tem o potencial de auxiliar na obtenção de economia e qualidade (melhoria). Usar componentes existentes e devidamente testados e documentados no desenvolvimento de um produto novo fornece, de certa forma, maior segurança, tanto da qualidade final do produto quanto na habilidade de realizar uma gestão adequada do processo de desenvolvimento. Permitindo, assim, maior realismo na estimativa de prazos e na alocação de recursos. O reuso de componentes também impacta na economia pela não necessidade de implementar código ou construir artefatos já desenvolvidos em outros projetos e pelo uso de artefatos mais amplamente testados, por já terem sido utilizados em outros projetos.

Visando prover reuso, neste trabalho é utilizado o conceito de repositórios de

artefatos de software como sendo, essencialmente, sistemas de banco de dados que armazenam artefatos e podem ser vistos como compostos por camadas. As superiores referindo-se à aplicação e às interfaces responsáveis pelo gerenciamento dos objetos e relacionamentos entre eles e a recuperação dos artefatos, entre outras possíveis funcionalidades. Na sequência, há o sistema do repositório em si, que contém os metadados dos artefatos. Os metadados e o próprio conteúdo dos artefatos, quando editáveis, são utilizados na busca dos artefatos. Subjacente ao repositório está a camada persistente, contendo o banco de dados responsável por armazenar os artefatos.

Porém, mais do que possibilitar o reuso, é desejável que a recuperação dos artefatos seja simples e conveniente, no sentido de possibilitar a recuperação entre os artefatos armazenados dos que melhor atendem aos critérios de busca. Para isso, é vital que o armazenamento dos artefatos esteja pelo menos, minimamente, estruturado. O ideal é que esteja muito bem estruturado, embora isso nem sempre seja possível pela diversidade de artefatos armazenados e pela cultura organizacional, conhecimento e interesses das equipes de projeto que podem influenciar na caracterização dos artefatos no momento do armazenamento.

O repositório que será utilizado para aplicar o algoritmo de busca desenvolvido como resultado final deste trabalho foi implementado como trabalho de conclusão de curso por Adriana Ariati (ARIATI, 2012). Adriana implementou a busca utilizando somente instruções *Structured Query Language* (SQL). Essas instruções são baseadas na existência ou não de palavras-chave que fazem parte dos metadados dos artefatos, que são utilizadas para caracterizar os artefatos e indicadas como critério de busca. Duas formas de busca foram adicionalmente implementadas. Uma delas utilizando lógica *fuzzy* (SANDI, 2013) e outra com o emprego de algoritmos genéticos (LUCION, 2013). Nessas duas propostas, às palavras-chave é atribuída a importância (em uma escala de 1 a 10) que é considerado que a referida palavra possui para o artefato a que está vinculada quando do seu armazenamento ou que é desejado a mesma tenha, na busca, como caracterização do artefato.

Neste trabalho, para a localização de artefatos que atendam aos critérios indicados na busca será utilizada a técnica da Inteligência Artificial (IA) denominada Raciocínio Baseado em Casos (RBC). RBC é usada para resolver problemas por lembrar de situação passada que seja semelhante à situação atual e pelo reuso de informações e conhecimento dessa situação (KUMAR; RAJ, 2010).

No contexto deste trabalho, os casos são representados por um conjunto de artefatos relacionados que são necessários para desenvolver um projeto de software. Os requisitos do

projeto definem o problema e os artefatos necessários para implementar o referido projeto representam a solução, compondo, assim, um caso. Para isso, por meio da criação de um conjunto inicial de casos definidos pelo usuário, o sistema será capaz de avaliar cada nova entrada, verificando se ela pode ser descrita por um caso já existente. Se não é possível, um novo caso poderá ser criado e armazenado no repositório.

Main, Dillon e Simon (2001) indicam que RBC se aplica aos domínios nos quais há modelos que os descrevem, se as soluções dos problemas podem ser reusadas no sentido de haver similaridade entre problemas, se há benefícios em adaptar soluções passadas e se é possível obter os dados armazenados de casos passados. No contexto de projetos de desenvolvimento de software essas características se aplicam, inclusive, porque artefatos são reusados em projetos distintos. A adequada caracterização dos casos pode ser realizada por meio de um repositório que armazene os artefatos com metadados bem definidos.

1.2 DEFINIÇÃO DO PROBLEMA

O desenvolvimento de software tem sido caracterizado por atrasos, custos de projeto acima do planejamento e produtos que não atendem aos interesses e necessidades dos usuários. Dados do The Standisth Group (THE STANDISH..., 2009) indicam que apenas 34% dos projetos de software são desenvolvidos dentro do planejamento inicial, atendendo prazo, custo e funcionalidades. Em relatório de 2014 (THE STANDISH..., 2014), referindo-se a projetos de grande porte, consta que 52% dos projetos sofrem mudanças, 42% falham e apenas 6% obtém sucesso. Falhos são os projetos que ou são cancelados durante o desenvolvimento ou são implementados, mas não são usados.

O reuso de artefatos de software, sendo artefatos os itens de trabalho produzidos e utilizados durante o ciclo de vida de software, pode ser uma alternativa para minimizar os dados que indicam os elevados índices de fracasso no desenvolvimento de software. Isso pela redução de trabalho não produzindo artefatos anteriormente produzidos e pelo uso de artefatos mais amplamente testados (pelo uso em projetos anteriores) que, teoricamente, possuem maior qualidade.

Contudo, embora reuso possa ser visto como uma das soluções há, problemas na efetividade do seu uso. Dentre esses problemas estão os relacionados:

a) Ao armazenamento - a efetividade da busca está relacionada à correta caracterização dos artefatos.

b) À busca - a melhoria dos mecanismos de busca utilizando técnicas de Inteligência Artificial, por exemplo, e que esses mecanismos considerem o contexto de uso e a cultura organizacional.
c) À cultura organizacional – os profissionais da área de software não estão acostumados a desenvolver artefatos para reuso, desenvolver com reuso e a compartilhar artefatos e código que desenvolvem. E, ainda, nem sempre estão dispostos a despender tempo para documentar devidamente os artefatos que desenvolvem ou que armazenam em repositórios.

Os aspectos relacionados à necessidade de mudança de cultura organizacional não são considerados no escopo deste trabalho. Em termos de armazenamento, para este trabalho, é utilizado o repositório de artefatos de software implementado por Adriana Ariati (ARIATI, 2012) que é considerado atender o contexto e escopo desta proposta. Assim, o problema tratado neste trabalho se refere à recuperação, à busca, de artefatos armazenados nesse repositório visando promover reuso. E a solução proposta utiliza RBC como técnica de busca de artefatos para implementar projetos de software. Esses projetos são caracterizados como casos e o conjunto de artefatos utilizados para implementá-los, a solução desses casos.

1.3 OBJETIVOS

A seguir são apresentados o objetivo geral e os objetivos específicos deste trabalho.

1.3.1 Objetivo Geral

Propor a caracterização de casos, que são projetos de software, para implementar um mecanismo de busca utilizando a técnica de RBC visando identificar os artefatos necessários ao desenvolvimento de projetos de software.

1.3.2 Objetivos Específicos

Buscar o entendimento de RBC visando identificar a melhor caracterização dos casos que são projetos de desenvolvimento de software.

Buscar o entendimento sobre artefatos de software e reuso visando definir a melhor

caracterização dos casos por meio da definição de metadados para projetos e artefatos.

Elaborar os requisitos que caracterizam um caso que serão os metadados de um projeto de desenvolvimento de software e seus artefatos.

Definir os critérios de busca visando identificar conceitos e atributos e a respectiva importância na caracterização dos casos.

Propor e implementar um algoritmo para busca de artefatos de software utilizando RBC.

1.4 JUSTIFICATIVA

O reuso de artefatos de software visa economia de recursos para o processo de desenvolvimento, bem como melhoria da qualidade dos produtos de software. Uma vez que não é necessário implementar artefatos já desenvolvidos para uso em outros projetos, o tempo necessário para desenvolver o projeto será, provavelmente, reduzido. Assim, a prática de reuso acarreta economia de recursos. Já a melhoria da qualidade do produto final ocorre pelo fato de os artefatos em questão já terem sido anteriormente testados e, possivelmente, melhor documentados pelos seus desenvolvedores e pelos usuários de projetos.

A técnica selecionada para o desenvolvimento do trabalho, a RBC, visa atender à especificação da descoberta de casos semelhantes para que artefatos possam ser reusados no desenvolvimento de projetos de software.

A caracterização dos casos e os critérios da busca foram realizados com base nos metadados definidos para o projeto e os artefatos. A busca não será realizada no conteúdo dos artefatos pela diversidade de tipos de arquivos, ocasionando dificuldade ou impossibilidade de edição de determinados tipos desses arquivos. Além disso, há artefatos que podem ser bastante restritos em termos de informações obtidas pela pesquisa em seu conteúdo.

Os arquivos armazenados no repositório são arquivos de texto (documento de requisitos, componentes de software editáveis, por exemplo), imagens e figuras, que além dos formatos tradicionais como “.jpg”, podem ter o formato definido pela própria ferramenta como os editores de modelagem Astah, Rational Rose e Microsoft Visio, por exemplo. Além desses há os arquivos não editáveis como componentes executáveis de código. Os arquivos .pdf podem ter o acesso ao seu conteúdo impossibilitado. De componentes de código se não houver comentários explícitos ou formas conhecidas de identificação desses comentários e de

denominação dos identificadores, a busca em seu conteúdo pode ser infrutífera.

O conhecimento sobre projetos, que é decorrente da experiência na sua realização, ainda é considerado difícil de recuperar de forma adequada pelas empresas. Embora haja sistemas para realizar o gerenciamento de documentos e de conteúdo da empresa é, ainda, uma tarefa difícil recuperar experiências de conhecimento relacionado a projetos (MARTIN; EMMENEGGER; WILKE, 2013).

Considerando esses argumentos, na proposta deste trabalho a busca utilizando RBC será realizada com base nos metadados definidos para os artefatos e projetos armazenados no repositório. A caracterização dos artefatos por meio dos metadados pode representar aspectos da cultura organizacional ou mesmo das equipes envolvidas nos projetos. Isso no sentido de que palavras-chave distintas podem ser utilizadas para caracterizar um mesmo atributo de um artefato ou projeto, por exemplo. Para minimizar as possíveis perdas na busca em decorrência desse aspecto, sinônimos (palavras-chave distintas com o mesmo significado) serão considerados pelo mecanismo de busca.

Um sistema RBC geralmente se refere a um aplicativo de software que identifica a solução para um determinado problema por examinar descrições similares e problemas anteriormente resolvidos e suas soluções associadas. O problema é representado pelas características que descrevem um projeto de software. A solução é representada pela caracterização dos artefatos de software (descrita por meio de valores de atributos relacionados aos artefatos e respectivos pesos atribuídos a esses valores) utilizados para implementar projetos de software com características semelhantes. Justifica-se, assim, o emprego da técnica de RBC como forma de promover o reuso de artefatos de software.

1.5 ESTRUTURA DO TRABALHO

A estrutura do trabalho contém uma relação dos capítulos e uma descrição sucinta do que cada um deles contém. Na Seção 2 é apresentado o referencial teórico sobre raciocínio baseado em casos e estado da arte. E na Seção 3 está o referencial teórico sobre artefatos de software. A seção 4 apresenta o resultado deste projeto que se refere, essencialmente, à caracterização dos casos e a implementação da busca no repositório. O texto é finalizado com as considerações finais.

2 RACIOCÍNIO BASEADO EM CASOS

Este capítulo apresenta a fundamentação conceitual do algoritmo implementado utilizando a técnica de Raciocínio Baseado em Casos para indicar artefatos a serem utilizados na implementação de um projeto de software, provendo, assim, reuso de artefatos de software. O estado da arte também é apresentado.

2.1 CONTEXTO E CONCEITO DE RACIOCÍNIO BASEADO EM CASOS

O Raciocínio Baseado em Casos tem suas origens em pesquisas no campo da ciência cognitiva. É uma abordagem para resolver problemas tendo como base o conhecimento (GULFEM, 2011). É considerada por pesquisadores como Martin, Emmenegger e Wilke (2013) como uma metodologia com a qual humanos usam casos para resolver problemas e Kolodner (1992) complementa que ela também representa as formas que os humanos podem fazer as máquinas usar experiências passadas para resolver problemas. RBC foi inicialmente organizado a partir da modelagem da estrutura de recuperação de memória no cérebro humano e de como o mesmo é capaz de atingir possíveis soluções para novos problemas por meio da recuperação de memórias de problemas similares do passado (SCHANK, 1982 *apud* WATSON, 1999).

O princípio fundamental do RBC é o uso de experiências passadas na resolução de novos problemas (ZHANG; XIA; CAI, 2010, KANG; KRISHNASWAMY; ZASLAVSKY, 2014), ou como define Finnie e Sun (2003): problemas similares possuem soluções similares. Esse princípio fundamenta o conceito clássico de RBC proposto por Kolodner (1992) para quem Raciocínio Baseado em Casos significa usar experiências passadas para entender e resolver novos problemas. Para a autora, a essência do RBC consiste em usar casos anteriores para explicar situações novas, criticar soluções novas, raciocinar a partir de precedentes para explicar situações novas ou criar soluções equivalentes para um novo problema. Para Main, Dillon e Simon (2001), RBC é uma metodologia para resolver problemas pela utilização de experiências passadas. Essa metodologia envolve reter na memória problemas anteriores e suas soluções e, por referência a esses, resolver novos problemas.

Um caso pode ser conceituado como um registro de uma experiência prévia ou de um problema e sua respectiva solução (MAIN; DILLON; SIMON, 2001). Assim, em RBC cada

experiência contendo um problema e sua respectiva solução é considerada um caso e é armazenada em um banco específico (AAMODT; PLAZA, 1994, KANG; KRISHNASWAMY; ZASLAVSKY, 2014).

Para armazenar as experiências como casos, elas precisam ser caracterizadas. A resolução de problemas é baseada na recuperação de experiências similares por analogia entre os requisitos (critérios) de busca que caracterizam os casos e as características que caracterizam os casos armazenados. Os casos (experiências) recuperados devem ser analisados se eles resolvem adequadamente o novo problema ou se precisam ser adaptados. A adaptação pode resultar em um novo caso armazenado (GUO; ZHOU; LI, 2012). Considera-se que se os problemas apresentem um grau de similaridade adequado, a solução do problema antigo ou o raciocínio que levou a mesma poderá ser aproveitada, seja total ou parcialmente, para que o novo problema seja também resolvido (ZHANG; XIA; CAI, 2010).

As tarefas de sistemas RBC podem ser categorizadas em analíticas e de síntese (WANGENHEIM; WANGENHEIM, 2003). Nas analíticas geralmente um novo caso é comparado àqueles da base de casos para determinar a qual tipo ou classe pertence. A solução associada ao caso mais similar dentro da classe correspondente é então apresentada. A recuperação de casos é aplicada em classificação, diagnóstico, suporte de decisão, tutoriais, previsão e avaliação, entre outros. As tarefas de síntese visam criar uma nova solução complexa por meio da combinação de partes de soluções prévias. A descrição do problema é um requisito para a solução e o espaço das soluções é indefinido. Esse tipo de tarefa se aplica aos casos caracterizados como problema e solução ou problema e estratégia de solução e se aplicam à configuração, planejamento e projeto.

RBC auxilia a resolver problemas de domínios não familiares pela avaliação e modificação de soluções possíveis (aplicáveis), que são a saída de um raciocínio com certas condições e suposições de acordo com o histórico dos casos (WANG; LIN, 2010). Isso auxilia a encontrar a solução quando não há falta de descrição do problema, da modelagem ou de eficiência dos algoritmos disponíveis (RICHARD; EKARTA, 2010). RBC é uma estratégia eficiente e prática de resolver problemas (GUO; ZHOU; LI, 2012).

A determinação do uso ou não do RBC como a metodologia para a solução dos problemas encontrados pelo sistema em questão deve levar em consideração algumas características do conjunto usual de problemas, sendo as mais proeminentes (KOLODNER, 1992):

a) Os processos não podem ser aleatórios, sendo que os fatores que levam ao sucesso ou fracasso dos mesmos devem ser adequadamente representáveis em casos;

b) Sistemas sem casos excepcionais podem ser mais adequadamente modelados e apresentarão um conjunto de regras mais simples;

c) Em domínios que contém casos com baixo grau de similaridade, podemos ter casos que jamais serão reutilizados, eliminando assim qualquer valor em armazená-los;

d) A adaptação de soluções já existentes na base deve representar uma economia de recursos em relação a simplesmente criar novas soluções manualmente.

2.2 ELEMENTOS BÁSICOS DO RACIOCÍNIO BASEADO EM CASOS

A solução de problemas por RBC geralmente é proposta pelos diversos autores a partir de quatro etapas que definem os seus elementos básicos. Aamodt e Plaza (1994) definiram um modelo clássico para o ciclo do RBC, frequentemente referenciado como modelo 4Rs, que é constituído das partes: recuperar (*retrieve*), reutilizar (*reuse*), revisar (*revise*) e reter (*retain*). Hunt (1995) define também quatro fases, denominadas: recuperação (*retrieval*), adaptação (*adaptation*), avaliação (*evaluation*) e restauração (*repair*). Allen (1994) propõe as seguintes etapas: apresentação (*presentation*), recuperação (*retrieval*), adaptação (*adaptation*), validação (*validation*) e atualização (*update*). E Kolodner e Leake (1996) considerando um RBC como um processo de lembrar e adaptar ou lembrar e comparar propõe o ciclo RBC como composto por: recuperar (*retrieve*), adaptar (*adapte and justify*), criticar (*criticize*), avaliar (*evaluate*) e armazenar (*store*).

Os modelos propostos pelos diversos autores, basicamente, compartilham a ideia comum que processos de RBC consistem em recuperar, adaptar, avaliar e atualizar casos. Finnie e Sun (2003) destacam que esses modelos não separam o problema da solução, não satisfazendo o conceito básico de caso como sendo: caso = problema + solução (DUBOIS et al., 1999, FINNIE; WITTIG, 1998). E, ainda, que esses modelos ignoram a etapa de construção do caso base que é a principal tarefa do RBC. Assim, Finnie e Sun (2003) acrescentam ao modelo clássico de Aamodt e Plaza (1994) uma fase inicial denominada repartição (*repartition*). Esses autores consideram a construção dos casos uma tarefa muito importante para o RBC e o caso pode ser construído com base no particionamento do mundo possível de problemas e soluções. O particionamento depende de relações de similaridade que são encontradas nesse universo de problemas e soluções.

Na Figura 1 é apresentado o ciclo de RBC de acordo com as quatro fases proposta por Aamodt e Plaza (1994). Essas fases são utilizadas como base por diversos autores, com

denominações distintas, ou complementadas com outras fases como exposto no parágrafo anterior.

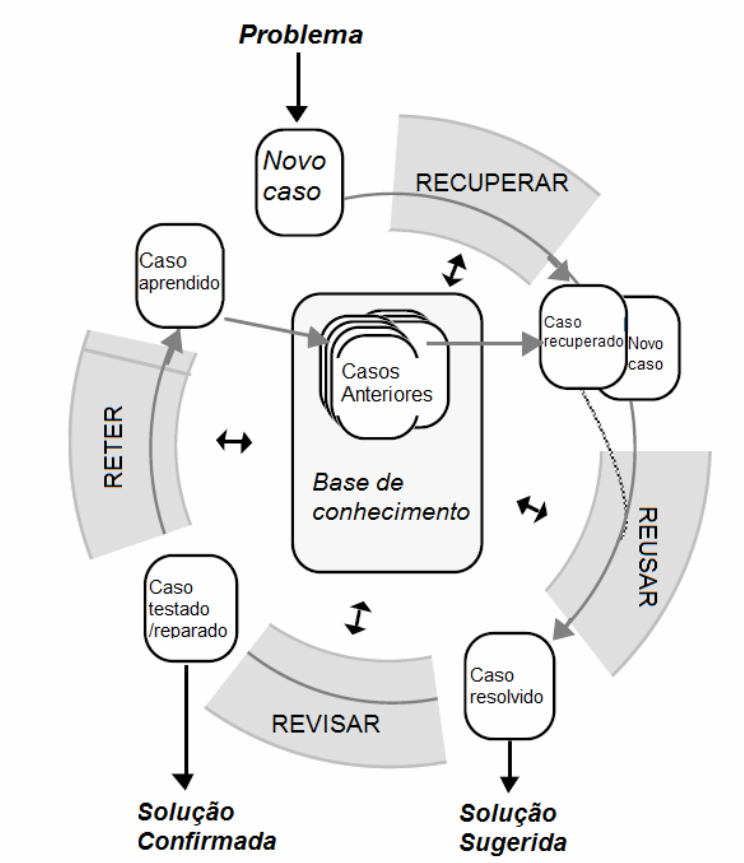

Figura 1 – Ciclo do RBC

Fonte: traduzido de Aamodt e Plaza (1994, p. 8).

Embora essas etapas possam ter identificações distintas, elas se resumem nos conteúdos apresentados a seguir, tendo como base Watson (2001), Wangenheim e Wangenheim (2003), Zhang, Xia e Cai (2010): representação do conhecimento, recuperação de casos, adaptação e aprendizado.

2.2.1 Representação do Caso

A representação do conhecimento é um aspecto fundamental da técnica denominada RBC. Um caso representa tipicamente a descrição de uma situação (problema) juntamente com as experiências adquiridas (solução) durante a sua resolução. Um caso é uma associação entre a descrição do problema e a respectiva solução.

Para que possam ser reutilizados, casos são organizados e armazenados em uma base de casos. A organização e a caracterização adequadas desses casos é fundamental para promover o reuso. Quanto melhor caracterizados estiverem os casos, mais efetiva poderá ser a busca e consequentemente o reuso apresentará maior potencial de sucesso. Contudo, ressalta-se que apesar da extrema importância de uma caracterização adequada para os casos, o processo de busca também depende da efetividade do mecanismo adotado.

Uma base de casos contém, geralmente, experiências que descrevem estratégias de solução que contribuíram para resolver o problema descrito, de forma que casos possam ser reutilizados. As experiências podem ser negativas, expressando tentativas que não obtiverem sucesso na solução de um problema. Essas experiências também podem ser armazenadas e consideradas (recuperadas) pelo mecanismo de busca na base de casos com o objetivo de indicar problemas potenciais e prevenir a repetição de erros passados.

A caracterização dos casos é baseada no conhecimento geral de domínio, que por sua vez é definido pelo vocabulário que contém os termos indicativos do domínio, bem como os sinônimos entre termos. O vocabulário pode ser acompanhado de um dicionário para o relacionamento de termos entre línguas diferentes (WANGENHEIM; WANGENHEIM, 2003).

Na caracterização dos casos a descrição do problema deve incluir toda a informação explicitamente considerada necessária para alcançar o seu objetivo específico e todo tipo de informação descritiva, que é normalmente utilizada para descrever casos daquele tipo. A descrição da solução contém os conceitos ou objetos usados para alcançar os objetivos específicos da tarefa realizada, considerando as restrições e a descrição da situação. Pode ser útil representar na descrição da solução (WANGENHEIM; WANGENHEIM, 2003): a solução em si; o conjunto de passos de raciocínio seguidos para resolver o problema; o conjunto de justificativas para as decisões tomadas durante a solução do problema; soluções alternativas aceitáveis que não foram escolhidas (e as respectivas razões e justificativas para sua exclusão); e expectativas acerca do que vai acontecer após a implementação da solução proposta.

Para Martin, Emmenegger e Wilke (2013) existem três principais abordagens para RBC e a decisão de qual utilizar deveria ser baseada no cenário da aplicação:

a) textual – os casos são armazenados como ou derivados de texto livre;
b) conversacional – a recuperação dos casos é realizada por meio de um diálogo;
c) estrutural – os casos são descritos utilizando um modelo de vocabulário ou domínio.

A proposta deste trabalho pode ser caracterizada como um modelo estrutural porque os atributos que caracterizam o caso e são utilizados como critérios de pesquisa.

2.2.1.1 Formas de Representação

Um dos grandes desafios do RBC é a busca da representação ótima de casos referentes a um dado domínio. Uma mesma entidade pode ser representada de formas diferentes em sistemas com objetivos distintos, uma vez que o RBC é uma técnica que permite que uma grande quantidade de tipos e relacionamentos de dados sejam empregados e que a porção de conteúdo descritivo relevante será dependente do domínio de aplicação. É necessário, também, considerar que a similaridade entre casos pode não ser evidente em um nível superficial, só emergindo após a realização de determinadas abstrações. Assim, é fundamental que os casos sejam adequadamente caracterizados e representados. Entretanto, isso não significa que cada objeto de estudo de um caso deva ser exaustivamente descrito. Assim, considerar o domínio de aplicação e o que é relevante ao mesmo é de suma importância. Portanto, é recomendado não usar mais descritores além dos que são estritamente necessários, para não afetar negativamente o desempenho do sistema (GEBHARDT et al., 1997), mas os necessários para caracterizar adequadamente o domínio.

Apesar de o RBC permitir a representação de casos vastamente distintos em aspectos fundamentais, como, por exemplo, a possibilidade de representar tanto casos estáticos quanto variantes no tempo, todo caso se refere a uma situação experimentada. E tal situação será invariavelmente constituída de um problema e de pelo menos uma solução. A descrição do problema é fundamentalmente constituída da meta, restrições e características. A meta é o objetivo geral que se procura atingir para a solução do problema, sendo que podem ser aplicadas algumas restrições sobre ela. As características são descritores que contemplam particularidades da situação (WANGENHEIM; WANGENHEIM, 2003).

A solução pode tanto conter uma descrição geral da ação que resolve o problema, quanto uma metodologia dos passos seguidos no raciocínio lógico do qual a resposta foi derivada. As decisões tomadas no processo de solução podem estar acompanhadas de justificativas para cada uma e também podem ser incluídas soluções alternativas.

Com os elementos básicos de um caso e os objetivos do sistema bem definidos, é necessário que se escolha um formato de representação. Apesar da flexibilidade do RBC, decisões tomadas no projeto podem limitar as opções viáveis. Deve-se considerar

primeiramente a linguagem que será utilizada na implementação, uma vez que a mesma pode limitar os tipos de dados disponíveis. Além disso, a seleção dos mecanismos de indexação e recuperação dos casos também afeta quais tipos de representação podem vir a ser mais eficientes. Por fim, há fatores como a possível preexistência de dados que precisem ser convertidos, os tipos de dados e estruturas que podem ser associadas aos casos do sistema, entre outros (MAIN; DILLON; SIMON, 2001).

Dentre as diversas formas de representação possíveis, se destacam as orientadas a objeto, redes semânticas, árvores k-d e, a mais comum, que é a representação atributo-valor. A descrição atributo-valor trata-se de um formato simples que prevê que cada descritor de um caso seja um par constituído de um atributo e seu respectivo valor (ou faixa de valores). O atributo estará associado a um tipo de dado fixo (como booleano ou inteiro) que se estenderá ao seu valor. O conjunto de pares atributo-valor pode ser tanto fixo para toda a base quanto particular a cada caso específico. Essa forma de representação simplifica a implementação de medidas de similaridades e é fácil de armazenar (por exemplo, em banco de dados relacionais), mas não é capaz de representar informação estrutural (WANGENHEIM; WANGENHEIM, 2003).

A representação orientada a objetos prevê que casos sejam representados como conjuntos de objetos, sendo esses caracterizados por um ou mais pares atributo-valor. Os objetos são estruturados por classes, que definem seus atributos e seu tipo. Já as classes são organizadas hierarquicamente, normalmente por meio de uma árvore, com as subclasses herdando as características de suas classes pai (WANGENHEIM; WANGENHEIM, 2003).

As redes semânticas são caracterizadas como um tipo de grafo que comumente ocasionam a representação de entidades como a composição de unidades de conhecimento (nós) e seus relacionamentos (arestas), resultando em uma estrutura de rede. As chamadas Redes de Recuperação de Casos (RRC) são resultantes de uma técnica que combina redes semânticas, RBC e elementos de redes neurais. Nessa técnica, entidades de informação (como pares atributo-valor) são representadas como nós em uma rede, estando ligados por arestas às instâncias dos casos aos quais estão relacionadas. Para a pesquisa na rede, emprestam-se conceitos de redes neurais, com a utilização de ativação propagada para recuperar casos similares na rede (LENZ; BURKHARD, 1996).

Árvores k-d são uma forma de árvore de busca binária estruturada com k dimensões, sendo k referente ao número de atributos armazenados na base. Diferentemente do que ocorre em uma árvore binária, cada nível corresponderá a uma das suas k chaves como critério de busca, de forma a encontrar um caso que possua um alto número de atributos similares aos

definidos no problema, sendo esse processo repetido recursivamente como uma pesquisa binária comum. Para a representação por árvores k-d, é necessário que se estruture a base de casos de forma que a mesma permita utilizar pesquisa binária. Para isso, se seleciona recursivamente algum atributo dos casos que permita dividir a base aproximadamente a metade, até que toda a base esteja estruturada dessa forma (WATSON; ABDULLAH, 1994).

Seja qual for a forma de representação selecionada para os casos, também é necessário que se determine a estrutura que governará o conjunto de casos. A eficiência da recuperação dependerá de uma boa estruturação. Arranjar os casos de forma plana e sequencial seria a forma mais comum, mas dependendo do tamanho da base as pesquisas podem se tornar ineficientes. Para resolver esse problema, existem modelos que utilizam uma disposição hierárquica de casos, agrupando os casos de maior semelhança previamente. Dentre os modelos de representação de casos, técnicas como árvores k-d e RRCs também operam como formas de organização (WANGENHEIM; WANGENHEIM, 2003).

2.2.2 Recuperação de Casos

A recuperação dos casos visa identificar o caso que melhor resolve um determinado problema ou obter casos úteis ou relevantes que podem ser utilizados com sucesso para resolver um problema (KANG; KRISHNASWAMY; ZASLAVSKY, 2014). A recuperação é a fase chave do CBR, uma vez que o sucesso desses sistemas é amplamente dependente do desempenho da recuperação (LOPEZ DE MANTARAS, et al., 2005). Contudo, ressalta-se que a caracterização dos casos também pode desempenhar papel crucial: quanto mais adequadamente e de forma mais completa os casos estiverem caracterizados mais efetivos podem ser os resultados dos mecanismos de busca que atuam sobre os dados que caracterizam os casos.

A medida de similaridade é uma forma de definir a pertinência dos critérios indicados na busca com a caracterização dos casos armazenados, ou seja, o quanto um ou mais casos armazenados na base de casos atende ao problema sendo descrito. A medida de similaridade é a formalização de uma maneira de julgamento de semelhança por meio de um modelo matemático concreto. No RBC o conceito de similaridade é formalizado de três formas (WANGENHEIM; WANGENHEIM, 2003): similaridade como predicado (uma relação entre objetos ou fatos, que existe ou não existe), similaridade como relação de preferência (similaridade maior ou menor) e similaridade como medida (a quantificação da extensão da

semelhança).

Wangenheim e Wangenheim (2003) ressaltam que a eficácia de enfoques baseados em casos depende essencialmente da escolha de um conceito de similaridade adequado para o domínio de aplicação e a estrutura de casos usados e que uma solução pode ser útil para um novo problema se ela: permitir solucionar o problema atual de alguma forma; evitar a repetição de um erro anterior; permitir solucionar o problema de forma eficiente, que exista, por exemplo, uma heurística passo a passo para calcular uma solução; oferecer a melhor solução para o problema de acordo com um critério de otimização definido; e oferecer ao usuário uma solução cuja lógica possa ser compreendida por ele.

A forma de recuperação de casos de implementação mais simples é a recuperação sequencial. Essa técnica consiste na comparação sucessiva do problema atual com cada caso da base. Os casos são então ordenados de acordo com os valores de similaridade resultantes, e uma quantidade predefinida dos mesmos é retornada. Trata-se de um método comprovadamente completo e, uma vez que cada caso será avaliado, não requer indexação. Entretanto, para bases de casos muito grandes, o desempenho será, provavelmente, ruim (WANGENHEIM; WANGENHEIM, 2003).

Árvores k-d e RRCs são, além de formas de representação de casos, também métodos de recuperação para os mesmos. Ambos os métodos têm alto nível de eficiência, porém requerem a construção de suas respectivas estruturas, o que significa certo custo computacional inicial. Além disso, são de mais difícil implementação do que um método sequencial (WANGENHEIM; WANGENHEIM, 2003).

Outra abordagem possível é a recuperação em dois níveis. Técnicas que utilizam essa estratégia visam em um primeiro momento reduzir o domínio de busca, realizando uma pré-seleção de candidatos. Essa fase ainda requer que todo o conjunto de casos seja comparado. Porém, a comparação seria mais simples e rápida, no nível mais baixo de representação sintática. Quaisquer casos que superem um valor de limiar de similaridade predefinido passam para a próxima fase. Nessa, o domínio reduzido de casos será mais uma vez comparado, dessa vez de forma aprofundada e completa (essencialmente sequencial), para que os casos possam ser ordenados e uma quantidade predefinida dos mais similares, retornados (FORBUS; GENTNER; LAW, 2010).

Por fim, há a recuperação orientada a índices. Trata-se de uma abordagem em duas fases. Inicialmente, uma estrutura de índices é gerada ou consultada pelo sistema da base de casos para cada tipo de busca possível. Isso permitiria acessar apenas os casos relevantes em uma segunda fase, de modo a executar a ordenação dos mesmos. Trata-se de um método com

alto custo de pré-processamento e manutenção das estruturas de índices (TANAKA; SUEDA, 1993).

2.2.2.1 Similaridade

A determinação do valor da similaridade entre casos tem o objetivo de selecionar os casos que possuem a maior probabilidade de conter a solução para o problema atual ou cuja solução possa ser mais facilmente adaptada ao objetivo. O grau da similaridade define a utilidade, no que diz respeito à reusabilidade, da solução (WANGENHEIM; WANGENHEIM, 2003).

O grau de similaridade é normalmente um valor numérico que representa o quão útil um caso é considerado ser para o reuso. Esse grau pode ser determinado a partir da comparação de apenas um atributo entre os casos, resultando na chamada similaridade local. Além disso, é também possível considerar todos os atributos dos casos e, assim, trabalhar com o caso como um todo, no processo conhecido como cálculo de similaridade global. Nessa técnica, também chamada de cálculo do vizinho mais próximo, considera-se que os casos estão distribuídos em um espaço multidimensional, com sua posição exata em relação à situação atual determinada pelo peso relativo de seus atributos. A soma ponderada desses atributos fornece a distância geométrica do problema e, portanto, os casos que apresentarem uma menor distância terão um maior grau de similaridade (WATSON, 1997).

A formalização do conceito de similaridade por meio de uma medida de distância é bastante difundida. Técnicas de cálculo de vizinho mais próximo são talvez as mais amplamente utilizadas em RBC, sendo empregadas pela maioria das ferramentas (WATSON, 1999). Em algoritmos que utilizam essas técnicas a similaridade do problema é determinada em relação a cada atributo dos casos armazenados na base de casos. Essa medida pode ser multiplicada por um fator de peso. Assim, a soma da similaridade de todos os atributos é calculada para fornecer a medida de similaridade entre os casos armazenados e a caracterização do caso que representa o problema para o qual se espera obter uma resposta. A resposta é definida em termos dos casos armazenados que resolvem o problema apresentado.

Uma vez que a medida de similaridade global é essencialmente a soma ponderada de todas as medidas de similaridade local, a mesma pode ser equacionada como um somatório de distâncias ponderadas, de forma a resultar em um valor numérico (Equação 1).

$$sim(Q, C) = \sum_{i=1}^{n} f(Q_i, C_i) \times w_i \quad (1)$$

Onde:

Q é o caso de entrada;

C é o caso da base;

n é o número de atributos de cada caso;

i é um atributo individual;

w é o peso dado ao atributo i;

f é a função de similaridade para o atributo i nos casos Q e C.

Normalmente o resultado deve ser entre zero (0) e um (1), sendo zero completamente não similar e um é exatamente similar.

Dentre as funções de cálculo de similaridade (vizinho mais próximo) estão a distância Euclidiana (Equação 2), a distância Euclidiana ponderada (Equação 3) e a distância de Manhattan (Equação 4), entre outras (DEZA; DEZA, 2009).

$$d(Q, C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \; sim(Q, C) = \sum_{i=1}^{n} f(Q_i, C_i) \times w_i \quad (2)$$

$$d(Q, C) = \sqrt{\sum_{i=1}^{n} w_i (q_i - c_i)^2} \; sim(Q, C) = \sum_{i=1}^{n} f(Q_i, C_i) \times w_i \quad (3)$$

$$d(Q, C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$

$$d(Q, C) = \sum_{i=1}^{n} |q_i - c_i| \; sim(Q, C) = \sum_{i=1}^{n} f(Q_i, C_i) \times w_i \quad (4)$$

Onde q e c são os atributos dos casos sendo comparados.

Para otimizar o desempenho da recuperação de casos relevantes ao problema atual, é possível que se definam atributos específicos para realizar a comparação entre um caso e a situação presente. Esses atributos são denominados índices. Assim, índices são combinações de seus atributos mais importantes, que permitem distingui-lo de outros e identificar casos úteis para uma determinada descrição do problema (WANGENHEIM; WANGENHEIM, 2003).

2.2.2.2 Indexação de Casos

Para que a recuperação de casos similares seja eficiente é necessário que além do sistema fornecer meios suficientes quanto à descrição do problema atual visando que a mesma seja coerente com a representação dos casos da base, também sejam definidas quais entidades de informação serão utilizadas no processo de recuperação. Para isso, usa-se a indexação de casos. Esse processo consiste em considerar certos atributos de cada caso como índices e, portanto, elementos passíveis de serem usados para comparação durante buscas.

A indexação de casos se refere à atribuição de índices para atributos que serão utilizados na recuperação desses casos e na comparação de casos recuperados como resultado de busca. A escolha dos índices é importante por possibilitar a recuperação dos casos mais adequados porque os índices determinam o contexto no qual o caso será futuramente recuperado (MAIN; DILLON; SIMON, 2001).

Assim, os índices devem representar boas abstrações do contexto do caso na base, permitindo que o caso seja recuperado em qualquer situação que ele seja de fato útil, mas não devem ser muito abstratos. Quando os índices de um caso são muito abstratos, o caso pode ser recuperado em muitas situações ou muito processamento será necessário para encontrar o caso adequado aos critérios de busca. Esses critérios caracterizam o problema para o qual se quer uma solução. Os índices devem refletir os aspectos importantes dos casos, os atributos que influenciam no resultado do caso e aqueles que descrevem as circunstâncias nas quais é esperado que o caso possa ser recuperado em buscas realizadas (MAIN; DILLON; SIMON, 2001).

A seleção de índices pode ser realizada tanto manualmente quanto via métodos automáticos. A escolha manual de índices dependerá da visão do especialista em relação à relevância de cada entidade de informação, uma vez que deve haver um caráter preditivo no exame realizado sobre o caso e suas possíveis aplicações, de forma que há alguma arbitrariedade. É importante que os atributos selecionados sejam capazes de descrever o objeto, buscando atingir a mínima representação possível que ainda seja capaz de manter a identidade do objeto. Também se deve considerar as características importantes para a solução e sua execução, bem como casos passados (KOLODNER, 1993).

Já os métodos automáticos operam normalmente das seguintes formas (KOLODNER, 1993):

a) Por meio de técnicas estatísticas pode-se procurar por atributos que tendem a ser

preditivos em todo o domínio;

b) Procurar por diferenças entre os casos, ou seja, índices com certa exclusividade;

c) Métodos de generalização que visam à criação de metacasos representando entidades de informação compartilhadas em um conjunto de casos;

d) Técnicas de aprendizado por indução podem realizar a extração automática de índices preditivos.

2.2.3 Adaptação

A fase de adaptação de casos recuperados visa ajustá-los aos requisitos da nova situação para que o problema seja resolvido. É o reuso do conhecimento armazenado, de forma adaptativa.

A partir do conjunto de casos recuperados podem ser realizados ajustes para adaptar a solução apresentada visando efetivamente resolver o problema. A adaptação procura por diferenças importantes entre os casos recuperados e o caso atual, aplicando fórmulas e regras que consideram essas diferenças na sugestão de uma solução (KUMAR; RAJ, 2010).

Segundo Aamodt e Plaza (1994), há duas principais formas de adaptação para casos: a adaptação transformacional, que prevê o reuso da solução em si; e a derivacional, que busca aplicar o método que resultou na mesma.

No reuso transformacional é assumido que a solução do caso antigo não será imediatamente a solução exata para o problema em questão. Com base nas diferenças entre os casos antigo e atual é possível que se determinem operadores que aplicados na solução do caso original possam resultar na solução para o problema. São chamados de operadores transformacionais (T) (AAMODT; PLAZA, 1994).

Já no reuso derivacional, consideram-se as informações referentes à como o caso anterior foi resolvido. Dentre essas há os operadores adotados, as justificativas para os mesmos, os objetivos, caminhos alternativos e falhos, entre outras. Esse processo então consistirá em seguir a metodologia de solução original como um plano para o novo contexto, primeiramente adotando os caminhos e práticas já comprovadamente bem-sucedidos (AAMODT; PLAZA, 1994).

Wilke e Bergmann (1998) adotam uma classificação mais extensiva. Além das estratégias de adaptação já mencionadas, consta também a adaptação nula, a composicional e a hierárquica.

Na adaptação nula, simplesmente não são realizadas mudanças nas soluções totais ou parciais retornadas pelo mecanismo de busca. Considera-se que quaisquer alterações ficarão a cargo do usuário. É um sistema de fácil implementação e, uma vez que é suficiente para aplicações simples, é a técnica adotada pela maioria dos sistemas comerciais (WILKE; BERGMANN, 1998).

A adaptação composicional prevê a construção de uma solução pela combinação de partes de outras soluções. É recomendado que a aplicação dessa técnica esteja acompanhada do armazenamento de soluções em um formato modular, ou seja, com componentes que possuam um certo grau de independência.

A adaptação hierárquica trabalha com casos descritos em múltiplos níveis de abstração. A técnica prevê que a solução inicial seja derivada do nível mais alto. Essa solução provisória é subsequentemente refinada conforme se percorrem os níveis de abstração. Não é necessário que apenas uma solução original seja utilizada no processo de obtenção de uma solução para o problema atual, sendo possível que soluções provenientes de casos diferentes sejam adotadas para cada nível de abstração, se forem julgadas mais adequadas.

2.2.4 Revisão e Aprendizado

Concluída a fase de adaptação é preciso que a solução encontrada seja avaliada, no processo chamado de revisão. Os problemas resolvidos com sucesso estão prontos para a retenção. Ou seja, eles e suas soluções serão armazenados como novos casos e passarão a integrar a base de conhecimento, podendo ser reutilizados.

O processo de revisão também pode indicar que a solução não está correta. Nesse caso é preciso que a solução seja reparada. O primeiro passo é a detecção dos erros e a recuperação ou geração de explicações para os mesmos. O relatório de erros pode ser incorporado à base de conhecimento com o propósito de ser aplicado em fases de reuso futuras, talvez prevenindo que erros similares venham a ocorrer. Definida a explicação do que ocasionou os erros, pode-se modificar a solução de forma a evitar que os mesmos ocorram, sendo esse processo normalmente realizado por um módulo de reparos com acesso a conhecimento causal e de domínio. A solução reparada será novamente revisada e encaminhada para retenção no caso do sucesso ou para novas iterações do processo de reparo caso ainda não esteja correta (AAMODT; PLAZA, 1994).

2.3 ESTADO DA ARTE

O RBC, devido à sua natureza e estrutura fundamental, é uma abordagem natural quando o objetivo é o reuso. Assim, foi desenvolvida uma série de trabalhos que buscam empregar o RBC para reuso de recursos, testando diversas técnicas e visando melhorar o desempenho do processo.

Uma das abordagens para recuperação de casos em uma base é a que busca pelos k vizinhos mais próximos em similaridade (WATSON; MARIR, 1994). Apesar de ser uma abordagem bem-sucedida, o valor de k é subjetivo e até mesmo em uma mesma base pode não ser definido o mais eficiente para todas as situações, prejudicando o desempenho da busca. Outra forma é a aplicação de um limiar de dissimilaridade, que limitaria a quantidade de casos retornados de forma dinâmica. Porém, assim como no caso do valor de k, determinar um valor adequado para o limiar pode ser problemático (TSAI; CHIU, 2009).

Uma abordagem possível para a determinação do limiar de dissimilaridade proposta por Tsai e Chiu (2009) é via aplicação de inferência estatística, garantindo assim que o valor obtido tenha relevância estatística. Sendo a dissimilaridade uma variável contínua, pode-se determinar seu comportamento por meio de distribuição probabilística. A derivação da função de distribuição de probabilidade permite que se determine se a dissimilaridade entre dois casos é estatisticamente suficiente. Assumindo que a dissimilaridade pode ser modelada por mistura de distribuições Gaussianas, é possível que um algoritmo de maximização de expectativas (DEMPSTER; LAIRD; RUBIN, 1977) seja empregado para estimar os parâmetros de distribuição e assim permitir que a função de distribuição de probabilidade correspondente seja obtida.

Com base nessa relação, os autores Tsai e Chiu (2009) propõem um modelo para busca dos vizinhos mais próximos, que garante que os vizinhos com similaridade estatisticamente significativa serão recuperados. Quando um caso novo é incluído na base, o sistema calcula a dissimilaridade do mesmo com todos os demais presentes. Se a dissimilaridade resultante da distribuição de probabilidade for baixa o suficiente, o caso é recuperado. Dessa forma, se exclui a necessidade de intervenção humana na decisão de quantos casos devem ser recuperados, ou qual é o limiar de recuperação. Entretanto, a eficiência do algoritmo de maximização de estatísticas ainda é relativamente pobre (TSAI; CHIU, 2009).

Segundo Kang, Krishnaswamy e Zaslavsky (2014), sistemas que utilizam RBC geralmente consideram somente a informação de similaridade quando julgam quais casos devem ser recuperados. Uma vez que não há uma metodologia única e universal para a modelagem desse tipo de conhecimento, essa abordagem sofre de alguns problemas (LOPEZ DE MANTARAS et al., 2005). Primeiramente, requer um alto grau de intervenção de um especialista no domínio de aplicação e assim mesmo é um processo complexo e dificultoso (GUO; HU; PENG, 2011). Dessa forma, é comum que a similaridade seja imprecisa e subjetiva, não sendo igualmente eficiente para todos os problemas do domínio (PARK; CHOI; PARK, 2009).

Uma alternativa proposta pelos autores Kang, Krishnaswamy e Zaslavsky (2014) é a adição do conhecimento associativo. Esse representaria ligações evidentes e úteis entre quaisquer componentes dos casos da base. As associações emergem de um determinado conjunto de regras específicas, sendo que a relevância de cada regra depende do problema em questão. A aquisição de conhecimento associativo depende da mineração de evidências que apontem para a associação entre casos, sendo um processo não subjetivo. Uma vez que o conjunto de regras não é necessariamente fixo, esse processo é dinâmico, com o melhor conjunto sendo empregado em cada situação. Assim, a recuperação retornaria, além dos casos, regras de associação. Os resultados encontrados pelos autores indicam uma significativa melhora do desempenho em relação ao uso apenas do conhecimento de similaridade nas aplicações testadas (KANG; KRISHNASWAMY; ZASLAVSKY, 2014).

Uma vez que a eficácia da recuperação de casos para reuso é dependente da análise dos mesmos e suas relações, é conveniente que o conhecimento representado em cada base de dados seja estruturado de forma ontológica. Com uma semântica bem definida, o reuso seria facilitado tanto entre aplicações quanto organizações. Por meio da representação atributo-valor é possível estruturar os casos de maneira uniforme e possibilitar que conhecimento adicional não necessariamente contido em casos possa ser aproveitado (GARRIDO et al., 2008).

Garrido et al. (2008) propõem uma representação ontológica por meio de um metacaso, que contém a estrutura padrão dos casos. Cada domínio pode conter então instâncias particulares do metacaso. Essa estruturação, bem como o uso de sistemas multiagentes, permite que relacionamentos sejam facilmente estabelecidos e, portanto, que haja recuperação de dados de diferentes repositórios que usem a mesma arquitetura.

Outro trabalho com uma abordagem ontológica foi desenvolvido por Mendes, Girardi e Leite (2013). Quando se busca implementar um sistema de RBC agente, ou seja, que é capaz

de tomar decisões sem a intervenção do usuário em tempo real, o uso de ontologias permite que o sistema seja capaz de ter um conhecimento semântico mais aprofundado, de forma a recuperar casos com maior precisão.

Embora os trabalhos e propostas apresentados nesta seção utilizem tecnologias e metodologias distintas todos seguem, basicamente, os mesmos princípios, que são os que constam em Watson (1999):

a) Tentativas de resolver problemas pelo reuso de soluções de problemas passados.
b) A recuperação de problemas passados (casos) envolve definir a similaridade do problema com os casos armazenados.
c) Uma vez que um problema é resolvido, a sua solução é armazenada na biblioteca de casos como forma de armazenar a experiência na resolução de problema para reuso futuro.

3 ARTEFATOS E REPOSITÓRIO DE SOFTWARE

Neste capítulo está o referencial teórico do trabalho sobre artefatos e repositório de artefatos de software. As buscas realizadas pelo mecanismo de busca utilizando raciocínio baseado em casos ocorrem em um repositório de artefatos. Esse mecanismo visa promover reuso de artefatos de software.

3.1 ARTEFATOS DE SOFTWARE

O escopo e o contexto de desenvolvimento de software e de reuso considerados neste trabalho permitem propor a definição de artefato de software como os produtos e os resultados diretos e indiretos da realização das atividades de ciclo de vida de software, que possam ser armazenados.

A seguir são apresentados conceitos de artefato de software provenientes da literatura e embora possam ter visões e mesmo denominações diferentes fornecem suporte ao conceito proposto. Esses conceitos são apresentados visando fundamentar artefatos de software como objetos de reuso. Alguns conceitos relacionam artefatos exclusivamente aos elementos de código, enquanto outros são mais abrangentes corroborando com o conceito proposto.

Szyperski (1999), por exemplo, define componente de software como uma unidade de composição com interfaces contratualmente especificadas, tendo explícitas apenas as dependências de contexto. A definição deste autor se refere aos elementos ou componentes de código de software.

Voltados para reuso, estão os conceitos de Sametinger (1997), Lucredio (2006) e Ezran, Morisio e Tully (2002), apresentados a seguir. Esses conceitos são complementados pela especificação para ativos de software reutilizáveis proposta pela *Reusable Asset Specification* (RAS) (OBJECT..., 2005).

Artefato de software visto como componente é conceituado por Sametinger (1997) como parte do sistema de software que é identificável e reutilizável. Para Lucredio (2006) e Sametinger (1997) artefatos reutilizáveis são auto-contidos, explicitamente identificáveis, descrevem ou realizam funções específicas e têm interfaces bem definidas e documentação apropriada.

Ezran, Morisio e Tully (2002), definem artefato de software, com a denominação de ativo, como sendo qualquer artefato de software passível de reutilização, como,

documentação, padrões de análise e projeto, normas de codificação, dentre outros.

Em uma conceituação mais ampla, na especificação RAS (OBJECT..., 2005), artefatos são considerados como os produtos de trabalho originados do ciclo de vida de desenvolvimento de software. Esses produtos de trabalho são, por exemplo, documentos de requisitos, arquivos de código fonte, descrições de implantação e casos de teste, dentre outros.

Para a proposta pela *Object Management Group* (OMG) (OBJECT..., 2005) um ativo é composto por um conjunto de artefatos. Um ativo reutilizável provê uma solução para um problema em um determinado contexto. Um ativo possui regras de uso, um contexto de aplicação definido e pode ser customizado.

3.2 REUSO DE ARTEFATOS DE SOFTWARE

É comum o reuso estar associado à fase de implementação de software pelo uso de componentes ou elementos de código já desenvolvidos em outros projetos. Porém há outros elementos do ciclo de vida de software aos quais se aplica reuso. Na fase de requisitos, modelos de entrevistas e *checklists*, por exemplo, podem ser utilizados em projetos distintos. Na fase de análise e projeto, os diagramas de casos de uso, de entidade e relacionamento, de classes, planos e casos de testes e demais documentos podem ser reusados. Na implementação, além de elementos de código, boas práticas e padronização para documentação podem ser reutilizadas. Planos, procedimentos e casos de teste podem ser reusados para fase de testes.

O desenvolvimento de software pode ser visto como um processo fundamentado na criação e utilização de artefatos (KROLL; KRUCHTEN, 2003 *apud* SILVA; OLIVEIRA, 2009). Reuso se refere ao emprego de recursos já existentes para a solução de um novo problema (OBJECT, 2005). O reuso de artefatos de software tem como um dos seus objetivos: a localização de artefatos que possam ser reutilizados, o desenvolvimento de artefatos de acordo com uma estruturação que visa facilitar reuso, a obtenção de economia de recursos utilizados no desenvolvimento e uma melhor qualidade do produto final (JUSTO, 1996).

A adequada estruturação dos artefatos pode evitar problemas específicos como perda de atomicidade e consistência (TILLEY; HUANG, 2002), bem como dificuldades de versionamento. Esses fatores dificultam o reuso (SKENE; EMMERICH, 2006). Reuso de artefatos é um aspecto importante para o desenvolvimento de software (SOARES, 2004). A

Quantitative Software Management Associates relata que a utilização de componentes levou à redução de 70% no tempo do ciclo de desenvolvimento e de 84% no custo do projeto (LYCETT, 2011).

A economia de recursos está relacionada à não necessidade de desenvolvimento de artefatos que já foram desenvolvidos em outros projetos. A qualidade está relacionada ao uso de artefatos mais amplamente testados, visto que foram utilizados em projetos anteriores.

O reuso de artefatos pode ser dificultado em decorrência de aspectos culturais e padronizações distintas adotadas pelas empresas ou por equipes distintas em uma empresa ou profissionais distintos que pertencem a uma mesma equipe de projeto. Portanto, para facilitar o reuso é conveniente que seja utilizada uma abordagem ontológica, ou seja, uma modelagem que busque descrever entidades, classes, atributos e relacionamentos de forma explícita e com conceitos compartilhados, permitindo que tanto os usuários quanto a máquina sejam capazes de lidar com os dados (CHEN; CHEN, 2008).

Para Prieto-Diaz (1991), reuso está relacionado ao uso em situações novas de conceitos ou produtos previamente adquiridos ou construídos. Sendo que esses produtos devem ser definidos e/ou produzidos visando reuso, estarem armazenados de forma a facilitar a sua localização e a identificação de similaridade entre situações novas e antigas permitindo a adaptação e uso.

3.3 REPOSITÓRIO DE ARTEFATOS DE SOFTWARE

Repositórios são, essencialmente, sistemas de banco de dados que armazenam artefatos, bem como seus dados sobre esses artefatos, denominados metadados. Existem diferentes arquiteturas possíveis para sistemas de gerenciamento de repositórios, normalmente compostas por camadas. Uma delas é a arquitetura MOF (*Meta Object Facility*). As camadas superiores da MOF referem-se à aplicação e às interfaces responsáveis pelo gerenciamento dos objetos e relacionamentos entre eles, a sua recuperação, entre outras possíveis funcionalidades. Na sequência, há o sistema do repositório em si, que contém os metadados dos artefatos. Os metadados e o próprio conteúdo dos artefatos, quando editáveis, são utilizados na busca dos artefatos. Subjacente ao repositório se encontra a camada persistente, contendo o banco de dados responsável por armazenar os artefatos (PETROV; BUCHMANN, 2008).

Frakes (2005) ressalta que além dos metadados, os repositórios devem prover

mecanismos para certificação da qualidade dos componentes. Além disso, Frakes também ressalta a importância de os repositórios incluírem facilidades de gerência de configuração, provendo recursos para controle de mudanças e gerência de métricas de reuso.

3.3.1 Busca e Recuperação de Artefatos de Software

Shiva e Shala (2005) destacam que o maior desafio no reuso de componentes de software está no gerenciamento eficiente dos componentes visando prover localização e busca rápidas. A recuperação dos artefatos pode ser realizada baseada nos metadados, não sendo necessário realizar a busca no conteúdo do artefato. Mesmo porque o conteúdo do artefato pode não ser acessível, como ocorre com artefatos não editáveis.

A classificação de artefatos de software é vista como um meio de produzir uma organização sistemática (PRIETO-DIAZ; FREEMAN, 1987). Vocabulário, ou palavras predefinidas, pode ser empregado para categorizar e organizar sistematicamente artefatos de software. Um vocabulário pode ser controlado ou não controlado (VANDERLEI, 2006). Um vocabulário controlado é baseado em classificação ou palavras-chave que definem termos utilizados para especificar artefatos por meio de um conjunto predefinido de atributos. Um vocabulário não controlado possui a indexação realizada a partir de texto livre. A indexação pode ser automática e realizada a partir da recuperação dos termos mais relevantes presentes em um texto. O texto pode referir-se ao conteúdo do artefato ou aos campos de metadados.

Um método de classificação pode estar fundamentado em um ou em ambos os tipos de vocabulário, podendo ser, basicamente, classificação por enumeração, facetas ou palavras-chave (PRIETO-DIAZ; FREEMAN, 1987; FRAKES, 2005; VANDERLEI, 2006). Esses métodos de classificação são apresentados a seguir.

a) **Classificação por enumeração**

Os artefatos são organizados em classes hierárquicas mutuamente exclusivas. A classificação enumerativa tem como base uma estrutura hierárquica predefinida (VANDERLEI, 2006). Os métodos de indexação ou estruturação de repositórios podem ser divididos em duas categorias principais (SHIVA; SHALA, 2007): manual e automática. A enumeração é uma estrutura hierárquica de categorias e subcategorias, definindo uma árvore hierárquica.

b) Classificação baseada em palavras-chave

As palavras-chave podem ser extraídas automaticamente de textos contidos nos

próprios artefatos ou definidas por especialistas. Salton (1989) sugere que sejam definidos: pesos e frequências para distinguir a importância das palavras-chave; uma lista de palavras que não devem ser consideradas (as que não possuem importância na classificação dos artefatos, como preposições); sinônimos entre palavras; identificação dos radicais das palavras; e vínculo entre documentos relacionados.

c) Classificação por facetas

Uma faceta pode ser vista como uma propriedade do artefato que deve aceitar valores predefinidos (PRIETO-DIAZ, 1991). Uma faceta é um par atributo-valor, com vários valores predefinidos associados a cada atributo. Para Vanderlei (2006), os usuários classificam os componentes pela especificação de um ou mais valores para cada um dos atributos (facetas).

4 RESULTADOS

Este capítulo apresenta os resultados obtidos com a realização deste trabalho.

4.1 CONTEXTO

Em RBC, um caso é a abstração de uma experiência descrita por meio de atributos valorados que representam o conteúdo da experiência e o contexto de sua ocorrência. Para o repositório de artefatos, um caso é análogo a um projeto de software e se relaciona aos artefatos necessários para desenvolvê-lo ou implementá-lo. O caso é fruto de uma busca bem-sucedida ou de composição voluntária, e descreve o problema. Os artefatos necessários para implementar o projeto descrevem a sua solução. Compõe-se, assim, um caso como o conjunto que contém a descrição do projeto e os artefatos relacionados à sua implementação.

Os atributos que caracterizam o caso, que incluem as palavras-chave que caracterizam os projetos e os artefatos, são os seus metadados. A valoração dos atributos ocorre pelos valores atribuídos aos artefatos e pela importância associada a eles no cadastro dos projetos e dos artefatos e também na definição dos critérios de busca. O contexto é representado pelo conjunto de atributos associados e o grau de importância atribuído a cada atributo. O contexto é representado por uma descrição textual que complementa a caracterização realizada pelos atributos. O contexto pode ser utilizado para auxiliar a validar se a solução apresentada para o caso é adequada.

A representação do caso contém as informações que descrevem uma situação que tem impacto direto na sua solução. São as características do projeto de software que se quer desenvolver e para o qual são buscados artefatos armazenados no repositório. Casos representam experiências reais ou conhecimento que são relacionados às situações ou contextos específicos. Os projetos são exemplos do domínio do problema que é o reuso de artefatos no desenvolvimento de software. O reuso se torna importante neste contexto porque permite propor soluções em domínios não completamente conhecidos pelo gerente de projeto.

O algoritmo proposto, cuja caracterização dos casos é objeto deste trabalho, auxiliará a identificar artefatos necessários para o desenvolvimento de projetos de software. Nessa fase busca-se caracterizar os casos, no sentido de definir os metadados necessários. Os casos são caracterizados por:

a) Descrição do problema: a caracterização do que será resolvido. As características do projeto de desenvolvimento de software.

b) Descrição da solução: a solução utilizada para resolver o problema descrito. São os artefatos de software necessários para implementar o projeto descrito como problema.

c) Conclusão: verificar se a solução apresentada efetivamente resolve o problema. Se os artefatos sugeridos podem ser utilizados no desenvolvimento do projeto.

O algoritmo identificará uma solução para um problema atual por meio da identificação de casos semelhantes armazenados no banco de dados. Se uma nova solução é validada a mesma é armazenada no banco de dados e utilizada para resolver problemas futuros.

Nesse processo de busca utilizando RBC são identificadas três responsabilidades distintas:

a) Responsabilidades do sistema no registro – indexar as características relevantes por meio das quais será possível, futuramente, acessar o referido caso.

b) Responsabilidades do sistema na recuperação – sugerir soluções, notificar falhas (alerta) e auxiliar na crítica da solução.

c) Responsabilidade do especialista – escolher as características dos casos a serem considerados.

A Figura 2 representa uma visão geral e de alto nível da proposta deste trabalho. O uso de RBC tem como objetivo promover reuso, pela localização de casos semelhantes. Esses casos são projetos de software já implementados e os artefatos que foram utilizados nessa implementação. Esses artefatos estão cadastrados no repositório.

A composição de um caso pode ser realizada pelo especialista ou sugerida pelo sistema. Critérios de busca são utilizados visando identificar casos semelhantes ou para compor novos casos. A adaptação ocorre para a melhoria de um caso selecionado. Essa melhoria se refere à inclusão, substituição ou exclusão de artefatos relacionados a projetos de software (casos). A identificação de casos semelhantes é realizada a partir de critérios indicados na busca. Esses critérios caracterizam o projeto de desenvolvimento de software para o qual são buscados artefatos armazenados no repositório.

A ênfase deste trabalho está na descrição que representa a caracterização dos casos e na definição da metodologia para realizar a busca. Essa metodologia direciona a implementação do algoritmo, definido como mecanismo, de busca.

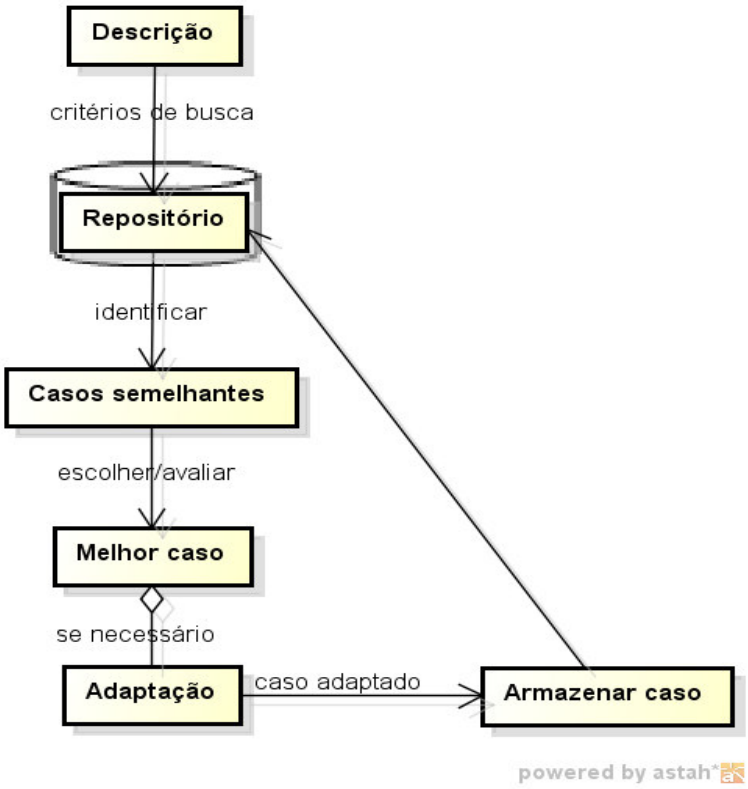

Figura 2 – Representação de alto nível da busca e composição de casos

Apresentação da Figura 2:

a) Descrição – definir as características que permitam a recuperação dos casos relevantes ao problema. É a caracterização do caso a ser localizado na base de dados ou repositório de artefatos. A descrição é representada pelas palavras-chave que caracterizam o projeto para o qual se quer obter os artefatos relacionados. Como resultado dessa caracterização espera-se especificar os artefatos a serem utilizados no desenvolvimento de um projeto de software. São os artefatos que podem ser utilizados para desenvolver o referido projeto.

b) Casos semelhantes – selecionar os melhores dentre os casos disponíveis. É a recuperação dos casos que melhor resolvem o problema que é representado pelos critérios de busca. A recuperação é realizada pela identificação das características mais significativas e que são comuns entre os casos. O mecanismo deve definir o grau de similaridade entre os casos e o novo problema ser proposto.

c) Melhor caso – escolher o caso que melhor atende aos critérios indicados na busca. O melhor caso permite a reutilização da solução associada ao caso.

d) Adaptação – é a estratégia para adaptar a solução ao melhor caso recuperado,

mesmo o melhor caso apresentado pode não resolver o problema por completo. Em termos de projeto, o contexto desta proposta, para adaptação são os artefatos que podem ser agregados aos selecionados, artefatos selecionados que podem ser substituídos ou mesmo excluídos para melhor atender aos requisitos do projeto (o problema) sendo representado. Um caso selecionado como atendendo ao problema pode ser adaptado, ajustado e a nova solução apresentada pode gerar um novo caso que comporá a base de casos. A adaptação pode envolver aprendizado quando um novo caso é armazenado na base de dados.

e) Armazenar novos casos – novos casos gerados a partir de adaptação de casos existentes, criados pelo especialista ou sugeridos pelo sistema são armazenados na base de casos. Os casos sugeridos pelo sistema devem ser validados pelo especialista.

Para determinar a similaridade entre os requisitos que descrevem o problema representado na busca e os casos armazenados no repositório que representam possíveis soluções para esse problema será realizado o seguinte:

a) Os atributos do projeto serão categorizados por meio de palavras-chave e da importância atribuída às mesmas. A importância define o peso das palavras-chave no cálculo de similaridade.
b) Na caracterização do problema para pesquisa os mesmos atributos serão valorados e importância atribuída. Esses atributos e importância serão utilizados como requisitos de busca.
c) Cálculo de similaridade local e global será realizado tendo como base as palavras-chave que descrevem o problema e os atributos dos casos da base.
d) Casos selecionados serão listados e ordenados pelo valor de similaridade obtido. Aqui poder-se-ia usar tanto um valor limite *k* para o número de casos recuperados, quanto estabelecer um limiar de dissimilaridade e recuperar todos os casos que o superem.

A representação do conhecimento do domínio inclui:

a) vocabulário – são as palavras-chave utilizadas para categorizar e definir os artefatos.

b) indicativo de termos sinônimos entre as palavras-chave visando, assim, atender a contexto de composição e de busca de casos.

A estrutura da proposta para a busca é representada pela Figura 3. Esse modelo de busca é organizado em três partes.

1) Conceito – são os metadados definidos para caracterizar os artefatos.

2) Atributos – são as características que cada conceito pode receber ao ser associado

ao artefato.

3) Valor do atributo – indica o valor das características de cada atributo, que pode ser usado para distinguir conceitos diferentes. O valor é associado ao artefato indicando a importância ou a pertinência do referido atributo para aquele conceito na caracterização do artefato.

Figura 3 – Representação do modelo de criação de novos casos

Os conceitos e seus respectivos atributos valorados são utilizados para caracterizar um projeto, um caso. Os conceitos se referem às palavras-chave atribuídas a um projeto.

Para que casos possam ser armazenados, a busca realizada e novos casos sejam gerados será necessário definir:

a) Os atributos relevantes para caracterizar o projeto, gerando um caso.

b) Definição dos índices – os índices indicam quais características do caso devem ser comparadas na busca. Os índices são inseridos nos casos no momento de sua inclusão na base de casos. Os índices auxiliam a organizar os casos de maneira que facilite a sua busca e recuperação. Na indexação também são atribuídos pesos às características dos casos, para que seja possível alcançar a recuperação de casos. No algoritmo de busca podem ser definidos pesos para as características mais importantes que normalmente são aquelas que possuem o maior número de ocorrências.

c) Definição dos métodos de recuperação para identificação de similaridades entre casos contidos na base e os requisitos definidos para o projeto (problema).

4.2 CARACTERIZAÇÃO DAS ENTIDADES DA BASE DE CASOS

Cada caso armazenado no repositório é um objeto que representa um projeto de desenvolvimento de software. Ele possui seus atributos particulares, descrevendo o problema em questão. A solução do caso é essencialmente o conjunto de artefatos empregados no desenvolvimento do software. Tais artefatos são entidades de informação também presentes na base de casos, tendo caracterização própria. Esses artefatos são adicionalmente ramificados por meio de diferentes versões. Tanto os casos quando os artefatos são primariamente

caracterizados por pares atributo-valor, na forma de palavras-chave e um grau de importância associado a elas.

Assim, a estrutura de caracterização de casos apresenta paralelos com os métodos de representação orientada a objetos. De fato, cada artefato específico é um objeto, podendo ser referenciado por múltiplos casos. Da mesma forma, as palavras-chave usadas para caracterizar tanto artefatos quanto casos também são entidades na base, relacionadas de forma ponderada aos objetos que descrevem. Assim, as entidades de informação que definem tanto o problema quanto a solução são de fato objetos relacionados de forma composicional ao caso. A Figura 4 exemplifica a estrutura de relacionamentos entre as entidades da base de casos.

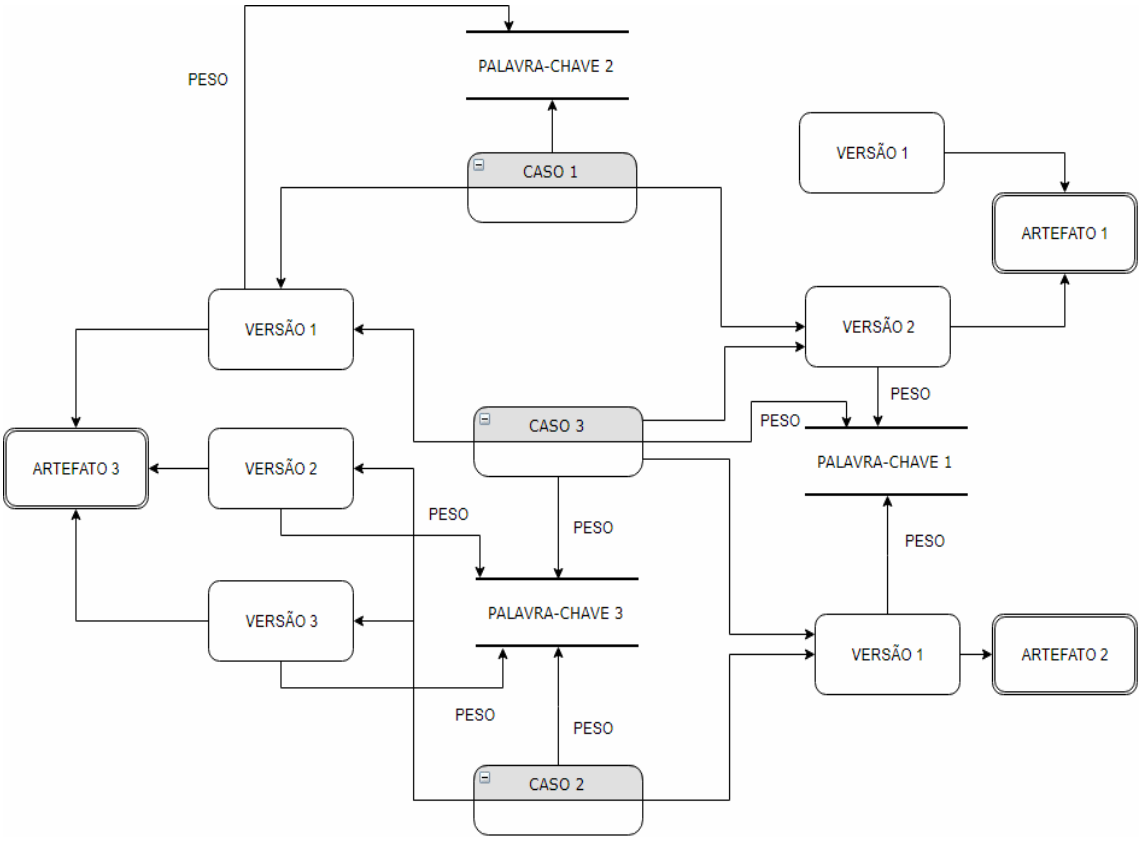

Figura 4 – Estrutura de relacionamentos entre os objetos da base

4.2.1 Caracterização dos Casos

Muito embora um caso descreva um projeto de software, é necessário que se estabeleça uma distinção entre as entidades caso e projeto no contexto do repositório. Apesar de um caso ser capaz de representar um projeto em sua totalidade, pode também ser usado

meramente para registrar o resultado de uma busca bem-sucedida. Além disso, a caracterização de um caso estimula o registro não apenas da solução final, mas também de outras abordagens e seus resultados. Para a manutenção da integridade do repositório e dos métodos de busca já implementados, optou-se pela coexistência dessas duas entidades, ao invés da substituição.

A caracterização dos casos envolve a descrição do problema e a documentação da solução. O problema é descrito por meio de requisitos que definem o projeto de software para o qual se busca uma solução semelhante armazenada como caso. A solução se refere aos artefatos de software que são utilizados para implementar o projeto descrito como problema.

No Quadro 1 é apresentada a caracterização do caso, composta pelos seus atributos que definem os metadados. Dentre esses metadados, as palavras-chave definem a representação essencial do problema.

Grupo	**Atributos**	**Descrição**
Identificação	Identidade	Identificador único de cada caso.
	Nome	Nome que identifica o caso.
Caracterização	Descrição	Detalhamento do problema. No caso de geração por busca, o campo será automaticamente preenchido pelos pares atributo-valor usados.
	Solução	Descrição das abordagens adotadas para a resolução do problema. Quando fruto de uma busca, apresentará a lista dos artefatos recuperados pelo RBC.
	Resultados	Apresentação dos resultados da aplicação de cada metodologia empregada na solução do problema. Para casos produzidos por busca, o campo será preenchido pelos artefatos selecionados pelo usuário.
	Categoria	Categorias a que o projeto pertence.
Relacionamentos	Palavras-chave	Palavras-chave utilizadas para caracterizar o caso. As palavras-chave são predefinidas (cadastradas). O relacionamento é ponderado por um grau de importância entre 0 e 10.
	Artefatos relacionados	Artefatos utilizados para implementar o processo. Os artefatos relacionados ao projeto de software representam a solução.

Quadro 1 – Metadados do caso

A Figura 5 apresenta a tela de registro de casos do repositório de artefatos.

Figura 5 – Tela de cadastro de casos

4.2.2 Caracterização dos Artefatos

No contexto do repositório, a entidade artefato atua como um meta-artefato, ou uma classe de objeto. Ela descreve um artefato ou tipo de artefato de forma geral. Os artefatos são devidamente individualizados na forma de versões, que representam os arquivos em si. A caracterização do artefato é definida por meio dos metadados utilizados no seu cadastro e no cadastro da sua versão. Desses metadados, as palavras-chave e seus respectivos graus de importância são utilizados pelo mecanismo de busca. Cada versão de um artefato conta com seu próprio conjunto de pares atributo-valor. Os outros metadados podem ser utilizados pelo especialista para validar a solução apresentada pelo mecanismo. Os artefatos da base já dispunham da opção de vínculo ponderado às palavras-chave. Assim, não foi necessária nenhuma adaptação nesse aspecto. A caracterização dos artefatos é apresentada no Quadro 2.

Grupo	**Atributo**	**Descrição**
Identificação	Identidade	Identificação única de cada artefato.
	Nome	Denominação atribuída ao artefato.
Caracterização	Descrição	Complemento ao nome do artefato. É um campo de busca para identificação do contexto e complemento aos metadados.
	Observação	Campo opcional para dados adicionais.
Relacionamentos	Versões	Implementações específicas do artefato.

Quadro 2 – Metadados de artefatos
Fonte: Adaptado de Ariati (2012, p. 34).

A Figura 6 exibe a tela de cadastro de artefatos.

Figura 6 – Tela de cadastro de artefatos

Enquanto o objeto artefato denota um meta-artefato, os objetos do tipo versão tratam da representação de implementações específicas, ou instâncias da classe artefato. Portanto, os arquivos e documentos do repositório são, de fato, representados como uma versão de um artefato. A caracterização de um artefato é geral e sucinta, mas versões são definidas por meio de um conjunto detalhado de metadados. De fato, as próprias palavras-chave estão atreladas às versões específicas, de modo a refletir quaisquer mudanças relevantes. A caracterização dos artefatos é apresentada no Quadro 3.

Grupo	**Atributo**	**Descrição**
Identificação	Identidade	Identificação única de cada versão de artefato.
	Nome	Denominação atribuída à versão.
	Local de disponibilização	Local de armazenamento caso o artefato não esteja no repositório.
Caracterização	Descrição	Complemento ao nome do artefato. É um campo de busca para identificação do contexto e complemento aos metadados.
	Categoria	Indica de componente, documento, procedimento ou outra.
	Requisitos para uso	Requisitos necessários para que o artefato possa ser utilizado.
	Histórico de alterações	As mudanças realizadas no artefato sejam melhorias, correções de erros ou acréscimos de funcionalidades.
Acesso	Disponibilidade	Define o tipo de acesso ao artefato: todos os usuários de repositório, grupos determinados ou somente o autor.
	Acesso ao conteúdo	Tipo de acesso ao conteúdo do artefato: leitura e

		escrita.
	Descrição da interface	Se componente de código a descrição da interface do mesmo.
	Tecnologia	Linguagem, ferramenta de desenvolvimento utilizadas no desenvolvimento do artefato.
Relacionamentos	Palavras-chave	Palavras-chave utilizadas para caracterizar o caso. As palavras-chave são predefinidas (cadastradas). O relacionamento é ponderado por um grau de importância entre 0 e 10.
	Artefato relacionado	Meta-artefato que descreve a versão em questão.
Qualidade	Testes de qualidade	Os testes de qualidade já realizados e os respectivos resultados obtidos.
	Aspectos de qualidade	Problemas identificados e restrições apresentadas. Solução dos problemas apresentados.

Quadro 3 – Metadados de versões de artefatos

Fonte: Adaptado de Ariati (2012, p. 34).

As Figuras 7 e 8 exibem as telas de cadastro de uma versão.

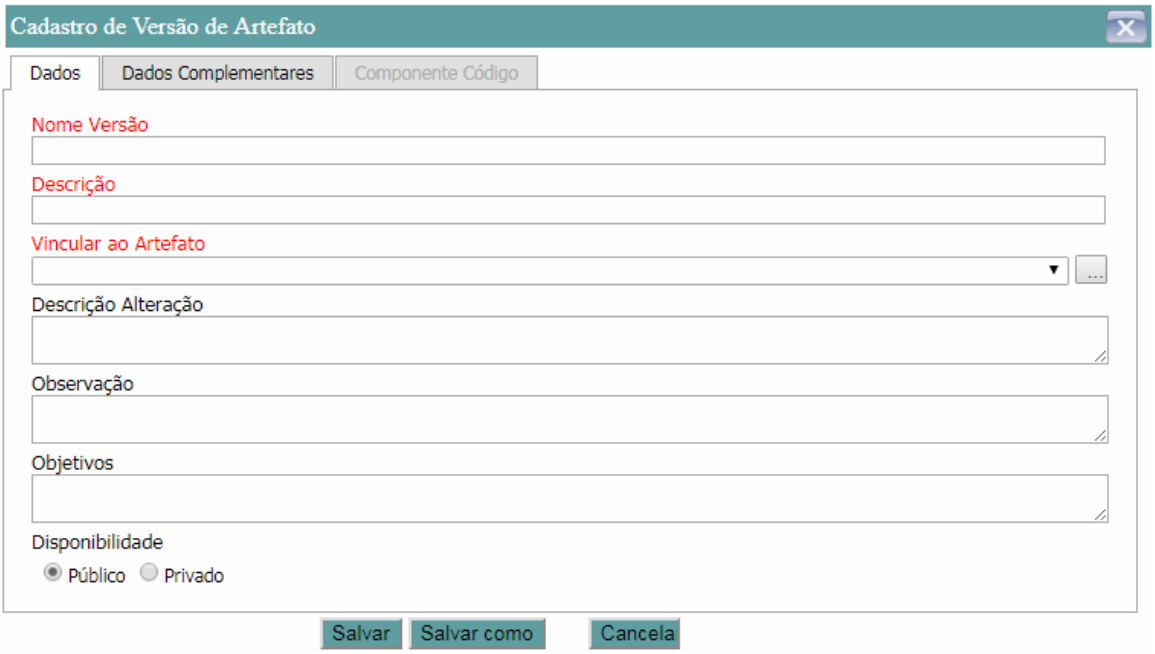

Figura 7 – Primeira tela de cadastro de versões de artefatos

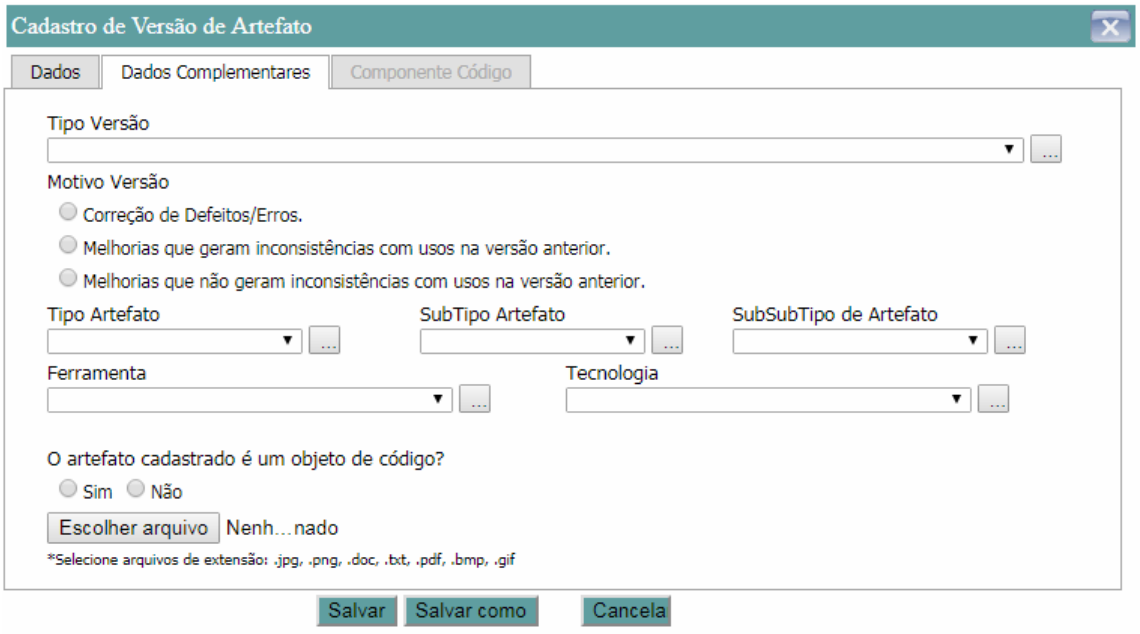

Figura 8 – Segunda tela de cadastro de versões de artefatos

Para a vinculação de artefatos aos casos, são consideradas versões específicas. Devido ao objetivo de reuso, é possível que outras versões de dado artefato não tenham utilidade ao usuário. E uma vez que cada versão é caracterizada por um conjunto de pares atributo-valor próprio, todas as versões pertinentes serão avaliadas pela busca. A Figura 9 apresenta a tela de vinculação de artefatos aos casos.

Figura 9 – Tela de vinculação de artefatos aos casos

4.2.3 Caracterização das Palavras-Chave

As palavras-chave têm importância crucial na base de casos, sendo os elementos responsáveis por criar a sinergia entre a caracterização e a recuperação dos casos. A caracterização de uma palavra-chave aparenta ser banal, consistindo apenas do campo de nome. Entretanto, deve-se considerar que as palavras-chave são utilizadas tanto para

caracterizar artefatos e casos como para a definição da busca via RBC. Assim, é fundamental que a base conte com um conjunto de palavras-chave suficiente para representar os possíveis problemas de seu escopo e que tais palavras-chave sejam adequadamente vinculadas aos objetos que descrevem. O processo de vinculação é simples, exigindo apenas a seleção do artefato ou caso em questão, da palavra-chave desejada, e de um grau de importância entre 0 e 10, que determinará o peso da aresta. As telas de criação de palavra-chave e de vinculação são apresentadas nas Figuras 10 e 11. A vinculação aos artefatos se dá de forma idêntica.

Figura 10 – Tela de cadastro de palavras-chave

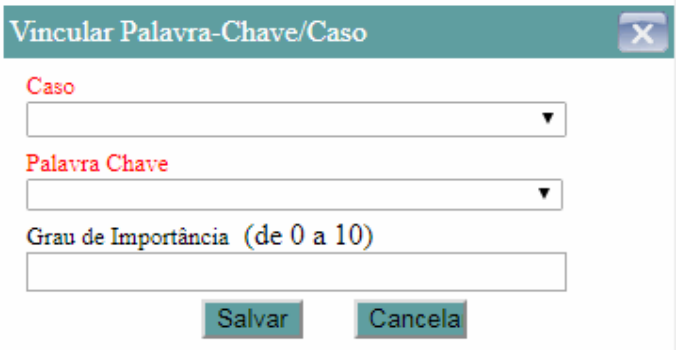

Figura 11 – Tela de vinculação de palavras-chave aos casos

4.3 RECUPERAÇÃO DOS CASOS NO REPOSITÓRIO DE ARTEFATOS

Considerando as formas de representação e organização dos casos na base, optou-se pela adoção de uma técnica de recuperação RBC em dois níveis. A fundamentação conceitual realizada neste trabalho levou a crer que RBC apresente boa efetividade como técnica de busca de artefatos para implementar projetos de software. Isso porque projetos de software e os artefatos necessários para implementá-los se caracterizam adequadamente como casos. E as palavras-chave utilizadas para caracterizar os projetos e os artefatos de software atuam perfeitamente como atributos valorados a serem considerados pelo algoritmo de busca. De fato, os relacionamentos entre essas palavras-chave e os objetos que descrevem são necessariamente ponderados, consolidando o par atributo-valor.

Enquanto a representação dos objetos da base tem similaridades com os métodos de representação orientada a objetos, as estruturas de organização e recuperação remetem às redes de recuperação de casos dirigidas a objetos. Os artefatos e casos são largamente descritos por relações com outras entidades de informação, e o método de recuperação percorre tais relações.

4.3.1 Implementação da Busca

A descrição dos dois níveis propostos:

a) Primeiro nível - a busca é realizada nos metadados dos casos e artefatos, especificamente nas suas palavras-chave. Devido ao uso prévio do repositório, há artefatos que não estão relacionados a quaisquer palavras-chave. Assim, há uma breve subfase inicial que consiste em selecionar apenas as entidades que possuem relação com alguma palavra-chave. Em seguida, são selecionados os objetos que possuem ao menos uma das palavras-chave que são indicadas como critério de busca, e que não estejam relacionados a palavras-chave proibidas pelo usuário. Dessa forma, são eliminados os objetos com elevado grau de dissimilaridade, reduzindo o domínio de busca para a fase seguinte. O objetivo desta fase é apenas selecionar os projetos que possuem palavras-chave coincidentes às indicadas na busca.

O trecho de código da Figura 12 apresenta as consultas SQL que correspondem à fase inicial do primeiro nível de recuperação de artefatos. Verificam-se os artefatos da base que têm relacionamento com pelo menos uma palavra-chave e se a palavra-chave em questão não está na lista das proibidas pelo usuário.

```
FbCommand cmd1 = new FbCommand("SELECT DISTINCT R.IDARTEFATO, A.NOME_ARTEFATO, V.IDVERSAO FROM "
             +"ARTEFATO A, RELACIONA_PC R, VERSAO_ARTEFATO V, PALAVRA_CHAVE P WHERE A.IDARTEFATO = "
             +"V.IDARTEFATO AND R.IDARTEFATO = A.IDARTEFATO AND P.IDPALAVRA_CHAVE = "
             +"R.IDPALAVRA_CHAVE AND V.IDVERSAO = R.IDVERSAO", conn);
            FbDataReader dr1 = null;
            try
            {
                conn.Open();
                dr1 = cmd1.ExecuteReader();
                while (dr1.Read())
                {
                FbCommand cmd2 = new FbCommand("SELECT DISTINCT R.IDARTEFATO, A.NOME_ARTEFATO, "
                                 +"R.IDPALAVRA_CHAVE, R.IMPORTANCIA, P.NOME_PALAVRA, V.IDVERSAO FROM "
                                 +"ARTEFATO A, RELACIONA_PC R, VERSAO_ARTEFATO V, PALAVRA_CHAVE P "
                                 +"WHERE A.IDARTEFATO = V.IDARTEFATO AND R.IDARTEFATO = A.IDARTEFATO "
                                 +"AND P.IDPALAVRA_CHAVE = R.IDPALAVRA_CHAVE AND V.IDVERSAO = "
                                 +"R.IDVERSAO "+nao, conn);
                                FbDataReader dr2 = null;
```

Figura 12 – Consultas SQL do primeiro nível de recuperação de artefatos

b) Segundo nível – a segunda fase de busca é realizada nos artefatos e casos recuperados na fase anterior. Para isso, é adotada uma busca sequencial, calculando a similaridade global entre cada objeto e os parâmetros de busca. Considera-se cada palavra-chave como uma dimensão em um plano multidimensional. O peso atribuído a cada palavra-chave na busca ou por seus relacionamentos atua como a posição de um ponto em tal dimensão. Assim, a similaridade entre a busca e um objeto recuperado pode ser quantificada por meio da distância geométrica entre suas ocorrências de pares atributo-valor. Para o cálculo da distância entre palavras-chave coincidentes, optou-se pela equação de distância Euclidiana, apresentada na Equação 2.

Em termos de algoritmo, a fase final do primeiro nível e a inicial do segundo estão conectadas. Os resultados da seleção anteriormente realizada são comparados ao conjunto de palavras-chave da busca para filtragem. A diferença entre a importância da palavra-chave para o artefato e para a busca é determinada e esse valor é inserido em uma tabela de armazenamento temporário juntamente com identificadores do artefato e da versão. O código que realiza essa ação é apresentado na Figura 13.

```
int indice = 0, pert = 0;
int cont1 = 0, cont2 = 1;
double DIST = 0;
for (int i = 0; i <= (possui.Length/2) - 1; i++)
{
        indice = i + cont1;
        pert = i + cont2;
        cont1++;
        cont2++;
        if ((possui[indice] == dr2.GetString(4)) && (dr2.GetString(0) == dr1.GetString(0)) &&
        (dr2.GetString(5) == dr1.GetString(2)))
        {
                DIST = Math.Pow(Double.Parse(possui[pert]) - Double.Parse(dr2.GetString(3)), 2);
                FbCommand cmdq = new FbCommand("INSERT INTO
        CRUZAMENTO(IDVERSAO,NOME_ARTEFATO,DIST) VALUES(@IDVERSAO, @NOME_ARTEFATO, @DIST)", conn);
                cmdq.Parameters.Add("@IDVERSAO", FbDbType.Integer).Value =
                Int32.Parse(dr2.GetString(5));
                cmdq.Parameters.Add("@NOME_ARTEFATO", FbDbType.VarChar).Value = dr1.GetString(1);
                cmdq.Parameters.Add("@DIST", FbDbType.Decimal).Value = DIST;
```

Figura 13 - Armazenamento de artefatos resultantes da busca (a)

Dessa forma, quanto menor a distância entre as instâncias de uma palavra-chave na busca e em um objeto retornado, maior será a similaridade. O processo é repetido para todas as palavras-chave coincidentes, agregando a distância encontrada a um somatório, e calculando o valor de distância final. Esse valor deve ser tratado de forma a exibir a similaridade de forma amigável ao usuário. Optou-se pela conversão da similaridade em uma porcentagem.

Para a normalização do valor, primeiro é determinada a máxima distância possível

entre dois objetos da base. Com essa finalidade, verifica-se a quantidade total de palavras-chave na base. Uma vez que os graus de importância estão no intervalo entre 0 e 10, a máxima distância em uma dada dimensão é 10. Portanto, é necessário que se aplique a equação de distância selecionada para um valor sempre igual a 10 e um número de dimensões que equivale ao total de palavras-chave. O resultado será a máxima distância possível no estado atual da base, que equivalerá a uma similaridade de 0%. Já a distância mínima sempre será 0, representada como uma similaridade de 100%.

Entretanto, mesmo para um número pequeno de palavras-chave na base, é improvável que qualquer objeto retornado se aproxime da maior distância possível. De fato, verificou-se que a similaridade dos objetos recuperados era sempre extremamente alta. Portanto, a fim de garantir uma representação com mais significado ao usuário, optou-se pela adoção de uma escala logarítmica, que impõe maior dificuldade à aquisição de elevado grau de similaridade. Assim, a Equação 5 foi empregada para a normalização da similaridade encontrada entre os parâmetros de busca e um objeto recuperado.

$$simil\% = 100\% - 100\% \times \log_{10}\left(9 \times \frac{dist_n}{dist_{max}} + 1\right) \qquad (5)$$

$$sim(Q, C) = \sum_{i=1}^{n} f(Q_i, C_i) \times w_i$$

Na implementação, isso se deu pelo colapso das diferentes palavras-chave pertinentes relacionadas a uma mesma versão de artefato em um único campo, somando os valores de distância. O cálculo de similaridade é efetuado de acordo com a Equação 5 e o valor resultante é testado contra o limiar de dissimilaridade, se houver. Os resultados são provisoriamente registrados na tabela que relacionará os artefatos ao possível novo caso a ser gerado. Uma implementação análoga à apresentada pelos segmentos de código foi utilizada para a recuperação de casos. A Figura 14 apresenta o código dessa implementação.

```
FbCommand cmdp = new FbCommand("SELECT SUM(DIST), IDVERSAO, NOME_ARTEFATO FROM CRUZAMENTO " +
            "GROUP BY IDVERSAO, NOME_ARTEFATO ORDER BY SUM(DIST)", conn);
FbDataReader drp = null;
        try
        {
        conn.Open();
         drp = cmdp.ExecuteReader();
         while (drp.Read())
         {
                Double y = 100 -
                100*Math.Log10(((9*Double.Parse(drp.GetString(0)))/Math.Sqrt(npc*100))+1);
                 if(y > x)
                 {
                     FbCommand cmdr = new FbCommand("INSERT INTO ARTEFATO_CASO_SOLUCAO
                (IDVERSAO,NOME_ARTEFATO,DIST) VALUES(@IDVERSAO, @NOME_ARTEFATO, @DIST)", conn);
                     cmdr.Parameters.Add("@IDVERSAO", FbDbType.Integer).Value =
                        Int32.Parse(drp.GetString(1));
                     cmdr.Parameters.Add("@NOME_ARTEFATO", FbDbType.Char).Value = drp.GetString(2);
                     cmdr.Parameters.Add("@DIST", FbDbType.Decimal).Value = y;
```

Figura 14 - Armazenamento de artefatos resultantes da busca (b)

Caso o usuário opte por determinar um limiar de dissimilaridade, objetos cuja similaridade seja inferior a tal valor são descartados. Por fim, o conjunto resultante de objetos deve ser apresentado ao usuário. Eles são distribuídos entre listagens de casos e artefatos, cada uma ordenada pelos respectivos valores percentuais de similaridade, de forma decrescente. A Figura 15 apresenta uma captura de tela de uma busca bem-sucedida.

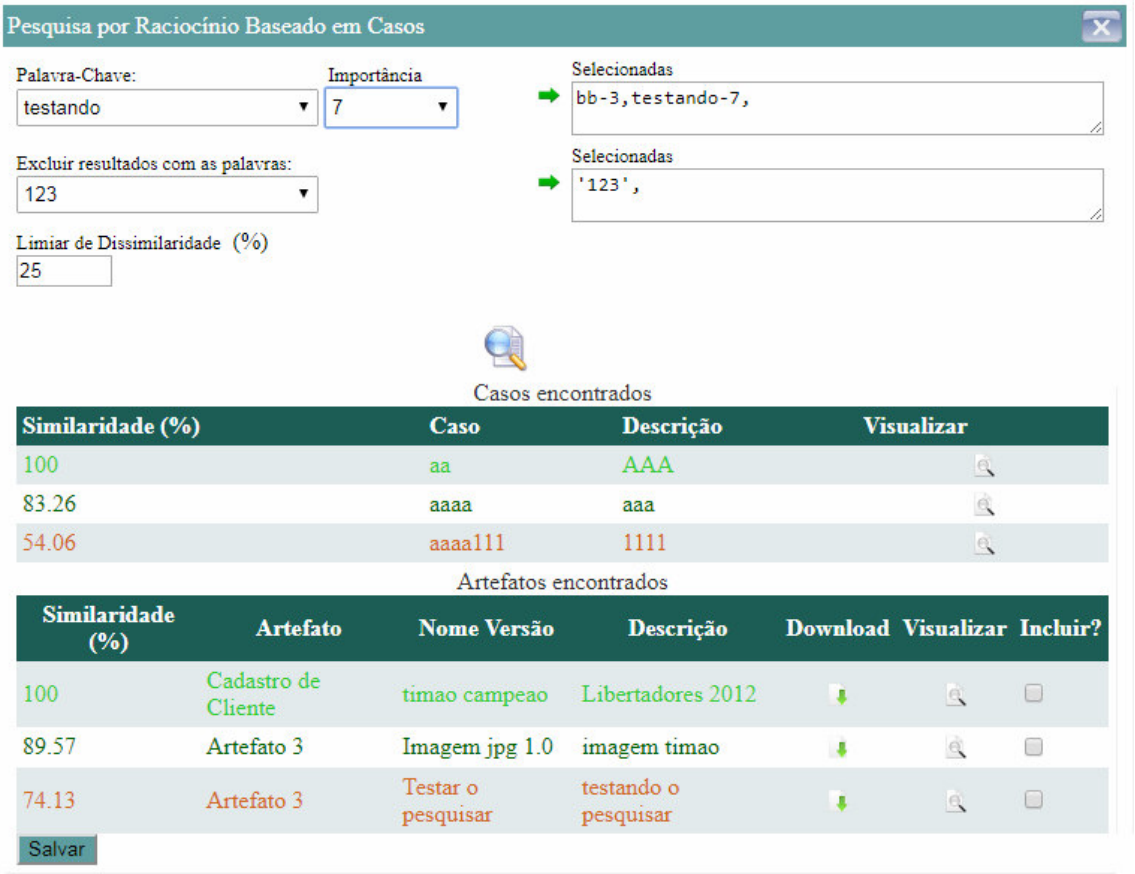

Figura 15 – Exemplo de resultados de uma busca

A listagem de casos apresenta metadados relevantes e a opção de visualização. No caso dos artefatos, são oferecidas, também, as opções de *download* e geração de novo caso. Na hipótese de o usuário considerar um ou mais artefatos como satisfatórios, ele tem a opção de selecionar os artefatos em questão e salvar como um novo caso. Esse é automaticamente preenchido por dados da busca. A descrição do problema consiste dos parâmetros de busca. A solução contém a listagem dos artefatos e as respectivas versões recuperadas pela busca. E o resultado lista as seleções do usuário, que também são automaticamente vinculadas ao caso. O usuário tem então a liberdade de editar o caso conforme julgar necessário.

4.3.2 Exemplo de Busca Típica

Esta subseção busca representar, em alto-nível, um processo de busca simples. O objetivo é ilustrar os relacionamentos entre as diferentes entidades de informação da base de casos. Será considerada apenas a recuperação de artefatos. Para que a busca seja realizada, o usuário deve caracterizar o problema por meio de um conjunto de palavras-chave e seus respectivos pesos. Uma caracterização adequada é facilitada por um banco de palavras-chave. A Figura 16 apresenta um exemplo de busca em uma pequena base, povoada por artefatos, suas versões, e as palavras-chave associadas. A busca, em azul, inclui três pares atributo-valor.

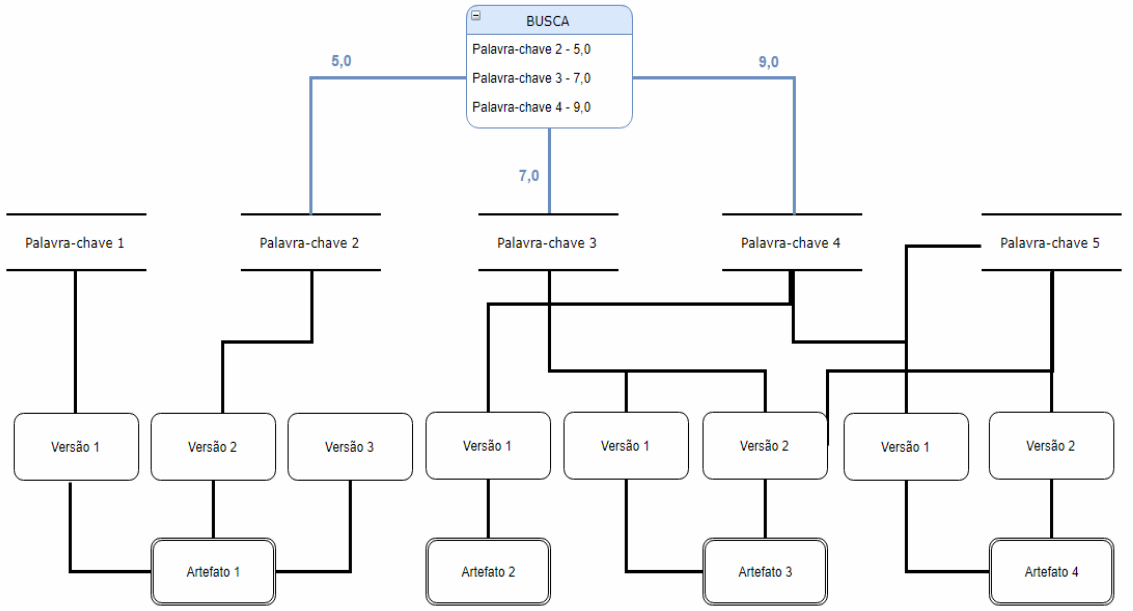

Figura 16 – Busca típica - caracterização do problema

Realizada a busca, serão primeiramente carregados todos os artefatos que estejam caracterizados por pelo menos uma das palavra-chave indicadas. A Figura 17 mostra, em vermelho, um artefato que não está devidamente caracterizado e que, portanto, não será mais considerado.

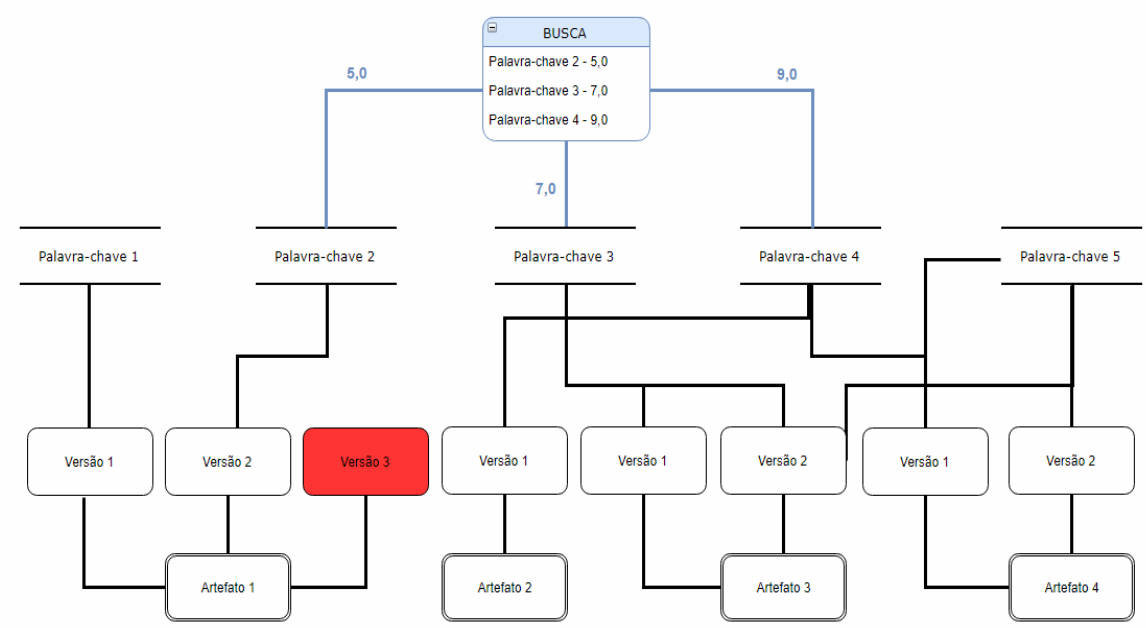

Figura 17 – Busca típica - eliminação de artefatos sem palavras-chave associadas

Em seguida, é iniciada a busca em dois níveis propriamente dita. Primeiramente, é selecionado o conjunto de artefatos que possuem ao menos uma das palavras-chave que compõem a busca. A Figura 18 exibe esse processo na forma do descarte de artefatos que estão descritos apenas por palavras-chave que não pertencem à busca, em vermelho.

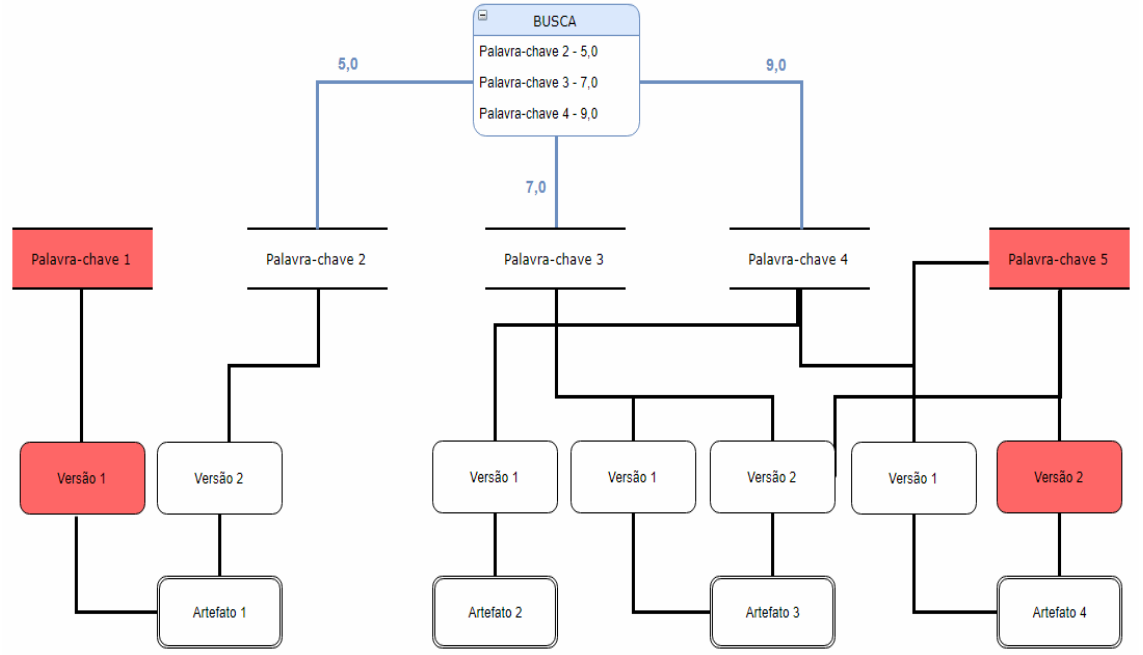

Figura 18 – Busca típica - eliminação de artefatos sem palavras-chave em comum com a busca

A Figura 19 ilustra o novo domínio reduzido de busca.

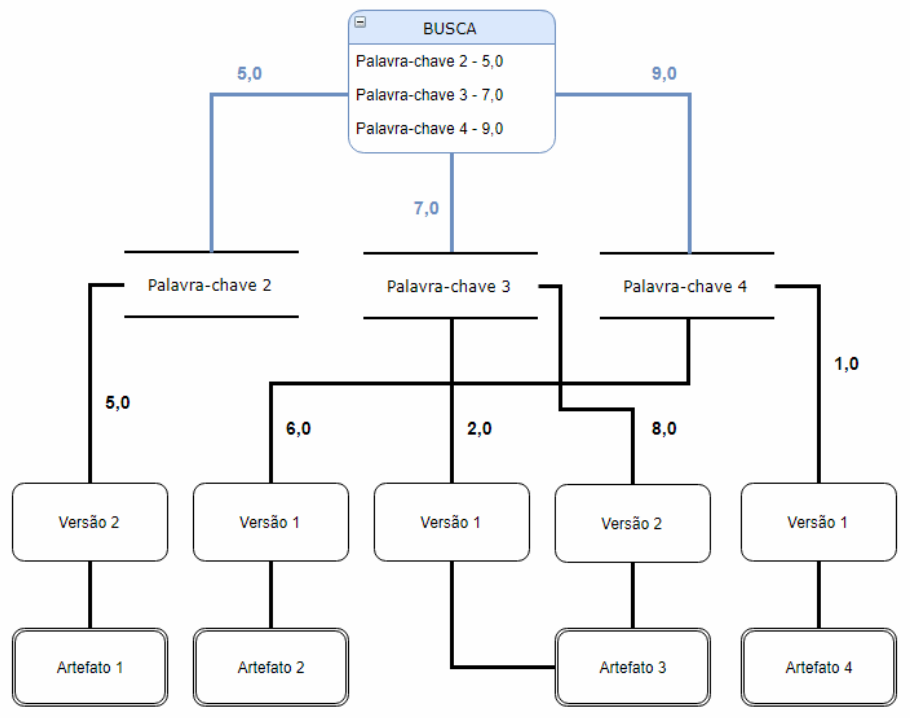

Figura 19 – Busca típica - redução do domínio de busca

Então, aplica-se o segundo nível da busca no domínio reduzido. O cálculo de similaridade global é realizado entre o conjunto de pares atributo-valor de cada artefato e o da busca. O valor da similaridade é tratado e convertido para uma porcentagem. Os artefatos são retornados ao usuário, ordenados pelo valor percentual. O usuário pode então visualizar os detalhes de cada artefato, bem como efetuar *download*. Por fim, o usuário pode selecionar os artefatos considerados úteis e gerar um novo caso. Por padrão, o caso será preenchido com dados gerados pela busca, mas o usuário está livre para editar o caso conforme julgar necessário. Na Figura 20, o usuário selecionaria os artefatos em verde, consolidando o sucesso da busca e gerando o caso representado na Figura 21.

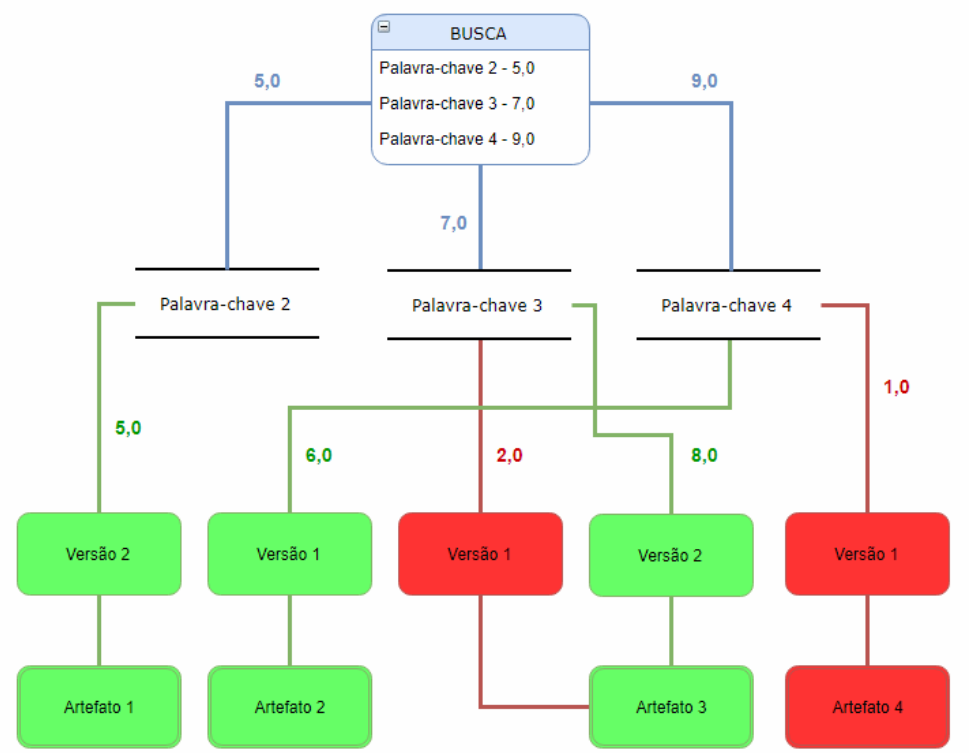

Figura 20 – Busca típica - seleção dos artefatos pertinentes

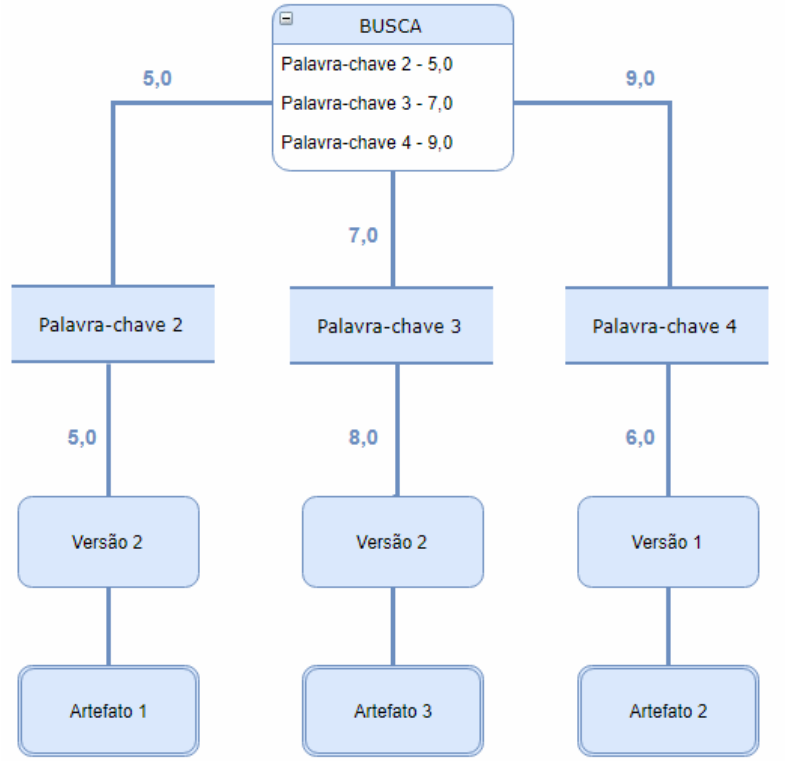

Figura 21 – Busca típica - geração de um novo caso

4.4 ADAPTAÇÃO E REVISÃO DOS CASOS

Devido ao domínio de aplicação do projeto, ou seja, desenvolvimento de software inviabiliza-se a implementação de uma fase de adaptação automatizada. Ficará a cargo do usuário a seleção de quais artefatos de software retornados são de fato aproveitáveis para o projeto. Também é responsabilidade do usuário a análise de quaisquer casos já compostos que sejam retornados pela busca. Posteriormente, o usuário tem a opção adicionar, substituir ou excluir artefatos da solução do seu problema. Então, uma vez que o sistema é responsável apenas pela recuperação dos objetos pertinentes, sem qualquer ação transformativa, trata-se de um método de adaptação nula por definição.

5 CONCLUSÃO

O reuso de artefatos de software tem entre os seus problemas a caracterização e a recuperação dos artefatos armazenados. Essas questões estão intimamente relacionadas, uma vez que metadados devidamente definidos contribuem para a efetividade dos mecanismos de busca. Propôs-se o uso da técnica de Inteligência Artificial denominada Raciocínio Baseado em Casos como um método de tratar ambos os problemas. Tomando como base as características do repositório de artefatos adotado, bem como a fundamentação teórica do RBC, buscou-se a definição de uma metodologia e a implementação dos mecanismos necessários para promover o reuso.

Considera-se cada projeto de software e seus artefatos relacionados como sendo componentes de um caso. A descrição do problema se dá por um conjunto de pares atributo-valor que representam o projeto de software. Os artefatos de software relacionados ao caso representam a solução para o problema. Os artefatos são entidades de informação na base de casos e como tais possuem sua própria caracterização em metadados, também na forma de pares atributo-valor. Esses são, por sua vez, concretizados como a relação entre palavras-chave e o grau de importância delas para o objeto em questão. Cada palavra-chave é também um objeto na base de casos. As arestas que unem as palavras-chave aos casos ou artefatos são ponderadas, denotando a importância da relação na caracterização do referido objeto.

Levando em consideração a representação dos casos, bem como a estrutura do repositório de artefatos, decidiu-se por um mecanismo de busca baseado em uma técnica de recuperação em dois níveis. Primeiramente, são excluídos quaisquer objetos que por ventura não estejam relacionados à pelo menos uma palavra-chave. Em seguida, são selecionados os casos que possuem uma ou mais das palavras-chave indicadas na busca, assim reduzindo o domínio de busca. Essa busca é realizada nos relacionamentos dos objetos da base. A fase seguinte trata da busca nos metadados dos artefatos e casos recuperados. A implementação dessa fase se dá de forma sequencial, com o cálculo de similaridade global sendo realizado entre cada objeto recuperado e o conjunto de dados de entrada. Assim, é possível a ordenação dos artefatos e casos retornados de acordo com o grau de pertinência ao problema descrito.

Foram implementados métodos para a limitação do escopo dos casos retornados. Em primeiro lugar, o usuário pode determinar palavras-chave que não devem estar presentes nos metadados dos artefatos a serem buscados. Ou seja, qualquer objeto que esteja relacionado a tais palavras-chave não será recuperado, independente dos demais metadados. Além disso, o usuário também pode estabelecer um limiar de dissimilaridade, na forma de um valor

percentual. Nenhum objeto cuja similaridade aos parâmetros de busca esteja abaixo do limiar será retornado.

Após a conclusão do processo de busca, um conjunto de soluções é apresentado ao usuário, consistindo de uma listagem de casos e uma listagem de artefatos, ambas ordenadas pelo grau de similaridade. Em relação aos casos, o usuário poderá visualizar detalhes e determinar se há algo aproveitável. Já quanto aos artefatos, há também a possibilidade de *download* e de composição de um novo caso. Essa se daria por meio da seleção dos artefatos recuperados que satisfazem o problema. O caso gerado é automaticamente preenchido por metadados que refletem todo o processo de busca, mas pode ser alterado conforme necessário.

Assim, se promove o reuso por meio da recuperação de artefatos que foram, de alguma forma, pertinentes a projetos de software no passado e que tem o potencial de demonstrar utilidade novamente. O ângulo de aprendizado, comum entre as técnicas de Inteligência Artificial, é explorado pela habilidade do sistema de busca em recuperar casos similares ao novo problema. Tais casos são frequentemente frutos de buscas já realizadas, uma vez que tal opção é oferecida para buscas bem-sucedidas. Dessa forma, o sistema tem o potencial de se tornar mais robusto e capaz de resolver problemas do seu escopo conforme ele é utilizado.

No que diz respeito a trabalhos futuros, a manutenção do repositório de artefatos seria a contribuição mais relevante. Em seu estado atual, ele exige um sistema operacional 32-bits e faz uso de versões de software consideravelmente datadas. Além disso, as contribuições por diferentes autores resultaram em um todo que peca pela falta de coesão. Assim, recomendar ia-se a atualização das tecnologias empregadas, a reforma do sistema para eliminar quaisquer conflitos e inconsistências, e a criação de uma metodologia para a padronização da codificação. Dessa forma, se criaria um ambiente propício para a comparação de desempenho entre os diferentes métodos de busca, bem como a eventual implementação de métodos adicionais.

Referente à implementação do RBC, há opções para o refinamento dos sistemas de caracterização e busca. Conforme discutido, o sistema tem paralelos com redes de recuperação de casos, em especial direcionadas a objetos. As RRCs, por sua vez, são inspiradas por redes neurais. Poder-se-ia formalizar tal classificação. Nesse caso, seria necessária a inclusão de mais tipos de relacionamentos, como o de relevância. Esse relacionamento, existindo entre objetos do mesmo tipo, permitiria a recuperação de objetos que de alguma forma são associados aos identificados pela busca. Além disso, tal relacionamento também permitiria uma interpretação menos literal das palavras-chave, como a associação de sinônimos ou outras formas de complementação automática de informação.

Dessa forma, poder-se-ia realizar recuperação por processo de ativação propagada. Essa teria tanto uma melhor capacidade de interpretar os parâmetros de busca quanto uma habilidade superior em identificar respostas pertinentes, mas não imediatamente óbvias.

As opções adotadas para o cálculo de similaridade, apesar de adequadas ao estado corrente do repositório, podem ser repensadas para uma eventual manutenção do desempenho. Por exemplo, uma opção de busca baseada em uma equação menos computacionalmente dispendiosa, como a de Manhattan, poderia ser oferecida em adição à atual similaridade Euclidiana. Entretanto, a escolha da equação de distância também impacta a representatividade dos atributos na ordenação dos objetos recuperados, de modo que ajustes dessa natureza devem ser feitos de forma cautelosa, isto é, não sem um amplo conjunto de testes que possam comparar a efetividade e o desempenho de cada uma das equações.

Durante a concepção do trabalho foi levantada uma possibilidade para a organização dos casos na base. Essa abordagem, de maior complexidade, seria o aproveitamento da estrutura organizacional que poderia emergir naturalmente na base de casos. Considerando-se uma distribuição em um plano multidimensional, com os casos arranjados com base nos pesos intrínsecos dos seus atributos, seria obtida uma estrutura com características que remetem a uma rede complexa, como, genericamente, ilustrado na Figura 22.

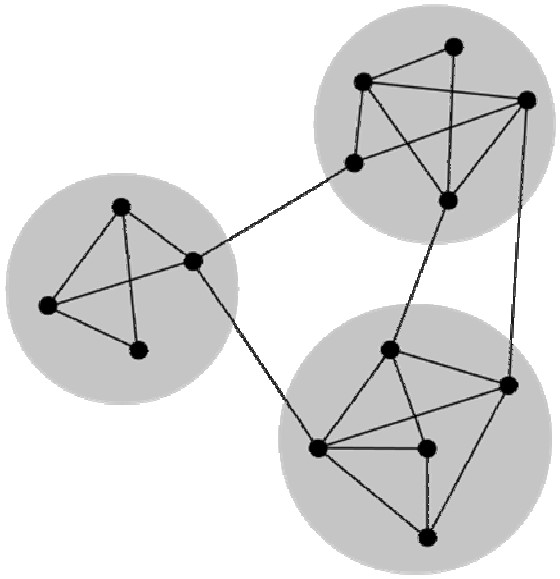

Figura 22 – Representação de uma rede complexa - comunidades

Nessa estrutura, de rede complexa com estrutura de comunidades, é bastante provável a formação natural de *clusters* de casos com grande similaridade entre si, especialmente após algumas iterações do processo de reuso terem acontecido na base. Tomando mais uma vez

redes complexas como uma estrutura análoga, tais agrupamentos de dados poderiam ser considerados como comunidades (áreas circulares na Figura 19). Essas comunidades poderiam ser representadas por metacasos.

Assim, postula-se que um algoritmo de extração de comunidades poderia organizar uma base de casos automaticamente, em categorias dinâmicas geradas por pertinência. Isso poderia também ser usado para reduzir o espaço de busca no processo de recuperação de casos. O algoritmo de extração de comunidades poderia ser relativamente simples, usando um cálculo de similaridade global e um valor fixo de limiar de dissimilaridade ou técnicas estatísticas para a delimitação das comunidades. Essas seriam então indexadas em metacasos contendo as informações relevantes para os casos internos. Dependendo da eficiência do algoritmo de identificação e representação de comunidades, o mesmo poderia ser preferível em bases que possuem uma grande quantidade de casos. A estrutura da rede poderia ser previamente construída e atualizada com cada nova adição à base, ou periodicamente.

REFERÊNCIAS

AAMODT, Agnar; PLAZA, Enric. Case-based reasoning: foundational issues, methodological variations, and system approaches. **AICom - Artificial Intelligence Communications**, IOS Press, v. 7, n. 1, 1994, p. 39-59.

ALLEN, Bradley P. Case-based reasoning: business applications. **Communications of the ACM**, v. 37, n. 3, 1994, p. 40-42.

ARIATI, Adriana. **Sistema web para armazenamento e recuperação de artefatos de software**. 2012. 111f. Trabalho de Conclusão de Curso (Graduação) – Curso de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná, Pato Branco, 2012.

CHEN, Ruey-Shun; CHEN, Duen-Kai. Apply ontology and agent technology to construct virtual observatory. **Expert Systems with Applications**, v. 34, n. 3, 2008, p. 2019-2028.

DEMPSTER, Arthur; LAIRD, Nan; RUBIN, Donald. Maximum likelihood from incomplete data via the EM algorithm. **Journal of the Royal Statistical Society**, Series B 39, p. 1–38, 1977.

DEZA, Elena; DEZA, Michel Marie. **Encyclopedia of Distances**. Springer, 2009.

DUBOIS, Didier; ESTEVA, Francesc; GARCIA, Pers; GODO, Lluís; LOPEZ DE MANTARAS, Ramon; PRADE, Henri. Case-based reasoning: a fuzzy approach. **Lecture Notes in Computer Science**, v. 1566, 1999, p. 79-90.

EZRAN, Michel; MORISIO, Maurizio; TULLY, Colin. **Practical software reuse**. Springer-Verlag London. 2002 Disponível em: <http://books.google.com.br/books?id=kIlpuK1GkLwC&pg=PA3&source=gbs_toc_r&cad=3#v=onepage&q&f=false >. Acesso em: 12 fev. 2014.

FINNIE, Gavin; SUN, Zhaohao. R5 model for case-based reasoning. **Knowledge-based systems**, n. 16, p. 59-65, 2003.

FINNIE, Gavin; WITTIG, Gerhard. **Intelligent support for internet marketing with case based reasoning**. In: Second Annual Collecter Conference on Electronic Commerce, 1998, p. 6-14.

FORBUS, Kenneth; GENTNER, Dedre; LAW, Keith. MAC/FAC: A model of similarity-based retrieval. **Cognitive Science: a Multidisciplinary Journal**, v. 19, n. 2, p. 141-205, 1994.

FRAKES, William B. Software reuse research: status and future. **IEEE Transactions on Software Engineering**, v. 31, n. 7, p. 529-536, 2005.

GARRIDO, José Luiz; HURTADO, Maria V.; NOGUERA, Manuel; ZURITA, Jose Mamuel. Using a CBR approach based on ontologies for recommendation and reuse of knowledge **sharing in decision making**. In: Eighth International Conference on Hybrid Intelligent Systems (HIS'08), 2008, p. 837-842, 2008.

GEBHARDT, Friedrich; VOß, Angi; GRÄTHER, Wolfgang; SCHMIDT-BELZ, Barbara. Reasoning with Complex Cases. **The Springer International Series in Engineering and Computer Science**, v. 393, p. 11-26, 1997.

GULFEM, Isiklar. An integrated case-based reasoning and MCDM system for web based tourism destination planning. **Expert Systems with Applications**, v. 3, n. 38, 2011, p. 2125-2132.

GUO, Sihai; ZHOU, Wenfeng; LI, Kaizhi. **Multi-layer Case-Based Reasoning approach of complex product system**. In: Third World Congress on Software Engineering (WCSE), 2012, p. 107-110.

GUO, Yuan; HU, Jie; PENG, Yinghong. Research on CBR system based on datamining, **Applied Software Computing**, v. 11, n. 8, p. 5006-5014, 2011.

HUNT, John. **Evolutionary case based design**. In: I.D. Waston (Ed.), Progress in Case-based Reasoning, LNAI 1020, Springer, Berlin, 1995, p. 17-31.

JUSTO, José L. B. **A repository to support requirement specifications reuse**. In: Information Systems Congress of New Zealand (ISCNZ'96), p. 53-62, 1996.

KOLODNER, Janet L. An introduction to case-based reasoning. **Artificial Intelligence Review**, n. 6, p. 3-34, 1992.

KOLODNER, Janet L. **Case-based reasoning**. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

KOLODNER, Janet; LEAKE, David B. **Case-Based Reasoning: experiences, lessons and future direction**. AAAI Press/MIT Press, Menlo Park, CA, 1996.

KUMAR, Suresh; RAJ, Dharm. **A contemporary approach to hybrid expert systems case base reasoning.** In: 2010 International Conference on Computer and Communication Technology (ICCCT), p. 736-740, 2010.

LENZ, Mario; BURKHARD, Hans-Dieter. Case retrieval nets: basic ideas and extension. In: KI-96: Advances in Artificial Intelligence. **Lecture Notes in Computer Science**, v. 1137, p. 227-239, 1996.

LOPEZ DE MANTARAS, Ramon; MCSHERRY, David; BRIDGE, Derek; LEAKE, David; SMYTH, Barry; CRAW, Susan; FALTINGS, Boi; MAHER, Mary L.; COX, Michael T.; FORBUS, Kenneth; KEANE, Mark; AAMODT, Agnar; WATSON, Ian. Retrieval, reuse, revision and retention in case-based reasoning. **Knowledge Engineering Review**, v. 20, n. 3, p. 215–240, 2005.

LUCION, Vagner; BORSOI, Beatriz T. Algoritmos genéticos como mecanismo de busca em repositório de artefatos de software. **Anais do XVIII Seminário de Iniciação Cientifica e Tecnológica**, Dois Vizinhos, 2013.

LUCRÉDIO, Daniel; FORTES, Renato Pontin de Mattos; ALMEIDA, Eduardo Santana; MEIRA, Silvio Lemos. **The Draco approach revisited: model-driven software reuse**. In: Workshop de Desenvolvimento Baseado em Componentes (VI WDBC), 2006. p. 72-79.

LYCETT, Mark. Understanding variation in component-based development: case findings from practice. **Information and Software Technology Journal**, v. 43, p. 203-213.

MAIN Julie; DILLON, Tharam; Shiu, SIMON. A tutorial on Case-Based Reasoning. **Soft Computing in Case Based Reasoning**, (Eds.) Sankar K Pal, Tharam Dillon and Daniel Yeung, Springer-Verlag (London) Ltd, 2001, p.1-28.

MARTIN, Andreas; EMMENEGGER, Sandro; WILKE, Gwendolin. **Integrating an enterprise architecture ontology in a Case-based Reasoning approach for project knowledge**. In: Enterprise Systems Conference (ES), 2013, p. 1-12.

MENDES, William; GIRARDI, Rosario; LEITE, Adriana. **Ontology-based architecture of a CBR agent**. In: 8th Iberian Conference on Information Systems and Technologies (CISTI), 2013, p.1-6.

OBJECT MANAGEMENT GROUP. **Reusable asset specification: documents associated with Reusable Asset Specification (RAS)**, version 2.2, 2005. Disponível em: <http://www.omg.org/spec/RAS/2.2/>. Acesso em: 24 abr. 2014.

PARK, Yoon-Joo; CHOI, Enmi; PARK, Soo-Hyun. Two-step filtering datamining method integrating case-based reasoning and rule induction. **Expert System with Applications**, v. 36, n. 1, p. 861-871, 2009.

PETROV, Ilia; BUCHMANN, Alejandro. **Architecture of OMG MOF-based repository systems**. In: International Organization for Information Integration and Web-based Application and Services (iiWAS2008), 2008, p. 193-200.

PRIETO-DIAZ, Ruben. Implementing faceted classification for software reuse**. Communications of the ACM**, v. 34, n. 5, p. 89-97, 1991.

PRIETO-DÍAZ, Ruben; FREEMAN, Peter. Classifying software for reusability. **IEEE Software**, v. 4, n. 1, p. 6-16, 1987.

RICHARD, Paul; EKARTA, Aniko. **Hierarchical case- based reasoning to support knitwear design**. CIRP Design Conference, 2010, v. 2, n. 4, p. 299-309.

SALTON, Gerard. **Automatic text processing: the transformation, analysis, and retrieval of information by computer**. Readings, Massachusetts: Addison-Wesley Pub, 1989.

SAMETINGER, Johannes. **Software engineering with reusable components**. Berlin Heidelberg: Springer-Verlag, 1997.

SANDI, Nathanyel; BORSOI, Beatriz T. Aplicação de lógica fuzzy em busca em repositório de artefatos de software. In: XVIII Seminário de Iniciação Cientifica e Tecnológica – SICITE, 2013, Dois Vizinhos. **Anais do Seminário de Iniciação Cientifica e Tecnológica**, 2013.

SHIVA, Sajjan G.; SHALA, Lubna A. **Software reuse: research and practice**. In: 4th IEEE Conference Information Technology (ITNG 07), IEEE CS Press, p. 603-609, 2007.

SILVA, Marcos; OLIVEIRA, Toacy; BASTOS, Ricardo. **Software artifact metamodel**. In: XXIII Brazilian Symposium on Software Engineering (SBES 2009), 2009, p.176-186.

SKENE, James; EMMERICH, Wolfgang. **Specifications, not meta-models**. In: 2006 International Workshop on Global Integrated Model Management (GaMMa '06), ACM, 2006, p. 47–54.

SOARES, Michel S. Comparação entre metodologias ágeis e tradicionais para o desenvolvimento de software. **Infocomp Journal of Computer Science,** v. 3, n. 2, 2004, p. 8-13.

SZYPERSKI, Clemens. **Component software**. Beyond object-oriented programming Harlow: Addison-Wesley, p. 411, 1999.

TANAKA, Toshikazu; SUEDA, Naomichi. Combining Strict Matching and Similarity Assessments for Retrieving Appropriate Cases Efficiently. In: P. Anick et al. (eds.), **Case-Based Reasoning and Information Retrieval.** AAAI Spring Symposium Series, 1993.

THE STANDISH GROUP INTERNATIONAL. **Big bang boom**. 2014. Disponível em: <http://blog.standishgroup.com/BigBangBoom.pdf>. Acesso em: 23 jun. 2014.

THE STANDISH GROUP INTERNATIONAL. **Chaos Report 2009**. 2009. Disponível em: <http://www1.standishgroup.com/newsroom/chaos_2009.php>. Acesso em: 28 abr. 2014.

TILLEY, Scott; HUANG, Shihong. **Documenting software systems with views III**: towards a task-oriented classification of program visualization techniques. In: 20th annual international conference on Computer documentation (SIGDOC '02). New York, NY, USA: ACM Press, 2002, p. 226-233.

TSAI, Chieh-Yuan; CHIU, Chuang-Cheng. **Developing a Significant Nearest Neighbor Search Method for Effective Case Retrieval in a CBR System**. In: International Association of Computer Science and Information Technology - Spring Conference (IACSITSC'09), 2009, p.262-266.

VANDERLEI, Taciana A.; GARCIA, Vinicius C.; ALMEIDA, Eduardo S.; MEIRA, Silvio R. L. **Folksonomy in a software component search engine – cooperative classification through shared metadata**. In: XX Simpósio Brasileiro de Engenharia de Software (SBES), 2006, p. 1-13.

WANG, Shen-Tsu; LIN. Wen-Tsann, Research on integrating different methods of neural Networks with case-based reasoning and rule-based system to infer causes of notebook computer breakdown. **Expert Systems with Applications**, 2010, v. 37, n. 6, p. 4544-4555.

WANGENHEIM Christiane Gresse von; WANGENHEIM Aldo von. **Raciocínio baseado em casos**. Curitiba: Editora Manole, 2003.

WATSON, Ian. A case study of maintenance of a commercially fielded case-based reasoning system. **Computational Intelligence: an International Journa**l, v.17, p. 387-398, 2001.

WATSON, Ian. **Applying case-based reasoning: techniques for enterprise systems**. Morgan Kaufmann, CA, USA, 1997.

WATSON, Ian. Case-based reasoning is a methodology not a technology. **Knowledge-Based Systems**, v. 12, p. 303-308, 1999.

WATSON, Ian; MARIR, Farhi. Case-based reasoning: a review. **Knowledge Engineering Review**, v. 9, p. 355-381, 1994.

WATSON, Ian; ABDULLAH, Salha. Developing case-based reasoning systems: a case study in diagnosing building defects. **Case based reasoning: prospects for applications (digest no. 1994/057)**. IEE Colloquium, p.1/1,1/3, 3-3, 1994.

WILKE, Wolfgang; BERGMANN, Ralph. **Techniques and knowledge used for adaptation during case-based problem solving**. In: 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, London, p. 497-506, 1998.

ZHANG, Xing; XIA, Huosong; CAI, Shuqin. **An decision-support system based on hybrid reasoning architecture**. In: IEEE International Conference on Information Management and Engineering (ICIME), Chengdu: IEEE, p. 335-339, 2010.