Resumo

Com um mercado cada vez mais competitivo e exigente, na indústria de software, as responsabilidades de um gerente de projetos aumentam cada vez mais. A Gerência de Projetos visa o sucesso de um projeto, este sucesso pode ser alcançado em diversos níveis, e.g. qualidade, preço, prazos, etc. Através de informações sobre experiências passadas é possível utilizar-se de métricas e estimativas como ferramentas de apoio no processo de Gerência de Projetos. Para tanto é preciso armazenar e organizar tais informações. Com demandas cada vez maiores de velocidade de entrega e qualidade no mercado de software, construir softwares torna-se, hoje em dia, um verdadeiro desafio para as organizações. Atualmente o sucesso de uma organização está diretamente ligado ao sucesso de seus projetos, portanto não é exagero dizer que a gestão de projetos pode ser encarada como gestão de negócio. O sucesso de um projeto pode ser afetado por algum evento capaz de causar impactos negativos. A área que trata destes eventos é a Gerência de Riscos que tem como objetivo minimizar estes impactos através da identificação de riscos, criação de planos de contingência, entre outras atividades, e o controle dos riscos evitando o insucesso. Com o objetivo de auxiliar na identificação de riscos de projetos, este trabalho propõe um modelo de identificação automática de riscos, na forma de um sistema de suporte à decisão, utilizando uma técnica de inteligência artificial chamada de Raciocínio Baseado em Casos (RBC) que se utiliza de informações sobre experiências passadas para resolver novos problemas.

Abstract

Nowadays, software market is a competitive environment with high quality demand where the project manager responsibility becomes bigger. The project management aims the project's success, which can be achieved in different levels, e.g. quality, cost, time, etc. Through past experience information it's possible to use metrics as support tool in the Project Management process, however to retain and organize this information is imperative. With the high demands of quality and shorter deadlines in the software development environment, to build software, nowadays, becomes a challenge to any organization. The organization's success is directly related to the success of its projects, therefore we can say that managing projects is also a way of managing business. A project success can be affected by some event capable of causing some negative impact. The Risk Management aims to minimize these negative impacts through risk identification, contingency plan, and other activities, and the risk control avoiding the non-success of the project. Intending to help in project's risk identification, this work proposes a model to automatically identify risks, by using an artificial intelligence technique called Case-Based Reasoning (CBR) that uses past experience information to help solving new problems.

Sumário

Índice de Figuras

Índice de Tabelas

Agradecimentos

Gostaria de agradecer a todos os professores do curso de Engenharia da Computação da UPE responsáveis pela minha formação como engenheiro, em especial a minha orientadora, Cristine Gusmão, sem a qual este trabalho não seria possível.

Agradeço também a todos os colegas de faculdade pelo companheirismo e ajuda durante minha graduação nesta instituição. Também não posso deixar de agradecer a Hugo Correia Lima e Conceição Oliveira pela companhia e apoio indispensáveis durante o desenvolvimento deste trabalho, vocês foram responsáveis pela finalização com sucesso desta monografia.

Faço aqui um agradecimento especial a Bruna Rafaela Ferreira pelo apoio, companhia, carinho e contribuição direta na conclusão deste documento através da elaboração de todas as imagens utilizadas.

Faço também um agradecimento mais do que especial a um amigo de todas as horas, Carlos Henrique Rocha Moreira, que tornou possível a realização deste trabalho, sendo diretamente responsável pelo sucesso alcançado pelo mesmo.

Por fim agradeço aos meus pais, pois sem seu apoio jamais teria chegado tão longe e finalmente concluído minha graduação.

Obrigado a todos.

Capítulo 1

Introdução

As organizações, atualmente, lidam com uma quantidade cada vez maior de informações. Com o avanço da Tecnologia da Informação, o armazenamento digital de documentos torna-se cada vez mais barato. Hoje em dia é comum encontrar organizações que lidam com diversos projetos e clientes ao mesmo tempo, tendo que, obrigatoriamente, lidar com uma grande quantidade de documentos e registros para controlar suas atividades.

Contudo, apenas guardar estas informações não é suficiente, é preciso uma metodologia para selecionar, salvar, manipular e recuperar documentos e registros, pois essas informações servem de apoio para a própria organização de forma que ela possa gerir seus projetos e atividades. Hoje em dia é comum a falta de estrutura das organizações para lidar com grandes quantidades de informações, e isto pode ter impacto direto nos projetos desenvolvidos pela mesma, bem como afetar a qualidade de seus produtos e serviços.

Este capítulo introdutório tem a finalidade de apresentar a motivação para o desenvolvimento deste trabalho (Seção 1.1), os objetivos definidos (Seção 1.2), a metodologia utilizada e a estrutura do documento, seções 1.3 e 1.4, respectivamente.

1.1 Motivação

Com o crescimento da indústria de software não apenas a quantidade de softwares existentes no mercado tem crescido, como também a qualidade exigida dos mesmos. Para produzir softwares de maior qualidade e capazes de executar tarefas cada vez mais complexas as organizações

envolvidas na construção de softwares têm investido na melhoria de seus processo de desenvolvimento de software.

Com isso, preocupações como manipulação das informações e histórico de trabalhos realizados passam a fazer parte do cotidiano das organizações que adotam processos bem estabelecidos de desenvolvimento de software.

Mesmo com processos de desenvolvimento maduros que ajudam a aumentar a qualidade do produto final, ainda é comum encontrar projetos que acabam falhando e trazendo diversos prejuízos às organizações envolvidas.

Em 1987, o PMI (Project Management Institute) publicou a primeira edição do guia PMBOK (Project Management Body of Knowledge) [1], um guia de boas práticas, onde encontramos abordagens e modelos para apoiar o gerenciamento de projetos. As práticas e metodologias encontradas no Guia PMBOK estão relacionadas a aspectos de projetos em geral, e.g. custo, prazo, qualidade e requisitos.

Porém, mesmo com a aplicação de boas práticas de gerenciamento um projeto ainda pode apresentar adversidades, estas podem ser situações ou eventos que geram um impacto negativo no projeto, podendo até fazer com que o mesmo falhe. A possibilidade da ocorrência dessa situação ou evento é considerada um risco de projeto. Estes riscos de projeto podem estar associados a diversos fatores do projeto, como qualidade, custo, tempo, requisitos, etc.

Todavia mesmo com a existência de um ou mais riscos de projeto, é possível gerenciá-los. No Guia PMBOK [1] podemos encontrar metodologias e práticas para abordar, planejar e executar atividades de gerenciamento de riscos. Dentre os processos da gerência de riscos estão: identificação de riscos, análise qualitativa de riscos, análise quantitativa de riscos, planejamento de resposta a riscos, monitoramento e controle dos riscos. Dentre esses processos, a identificação de riscos é crucial para que os mesmos possam ser gerenciados, sem a identificação dos riscos é impraticável a elaboração de ações para reduzir as ameaças aos objetivos do projeto.

Entretanto, além de identificar os riscos de projeto, é necessário documentar e armazenar as informações geradas de alguma forma, assim, caso o risco torne-se um evento ou situação, será possível recuperar as informações sobre as ações, anteriormente definidas, e colocá-las em prática. Ao final do projeto, estas informações, ainda estarão armazenadas. Logo a pergunta que surge é **como podemos utilizar informações de projetos passados para identificar riscos em um novo projeto?** Após uma investigação da literatura existente sobre ferramentas de auxílio à

identificação de riscos de projetos não foi encontrada nenhuma abordagem de Raciocínio Baseado em Casos, tomando como base experiência passada sobre projetos de software.

1.2 Objetivos

Este trabalho sugere que projetos de software semelhantes podem ser associados utilizando Raciocínio Baseado em Casos (RBC), e que através dessa associação é possível auxiliar a identificação dos riscos de um projeto de software baseando-se nos riscos, ocorridos ou não, de um projeto semelhante.

RBC é uma tecnologia de representação e processamento de conhecimento que usa a experiência passada para auxiliar na resolução de novos problemas [2]. A idéia é fazer uma analogia para descobrir se um problema é similar a outro já resolvido, aproveitando, desta forma, a solução aplicada anteriormente.

Dentro deste contexto, o principal objetivo deste trabalho é a construção de um modelo de recuperação de informações de projetos utilizando RBC. Os riscos de um novo projeto de software podem ser similares, ou até mesmo iguais, aos riscos de outro projeto de software semelhante desenvolvido no passado. Neste caso as ações preventivas e/ou corretivas executadas no mesmo podem ser utilizadas novamente, com as devidas adaptações necessárias.

Para o alcance deste objetivo, faz-se necessário, de forma específica:

1. **Estudo detalhado de técnicas de identificação de riscos** – Levantamento sobre técnicas e ferramentas utilizadas para identificar riscos de projeto;
2. **Estudo detalhado da técnica de RBC** – Consultar a literatura existente obtendo, desta forma, embasamento científico para este trabalho;
3. **Definição dos critérios de similaridade entre projetos** – Eleger quais características de um projeto de software serão usadas na comparação dos mesmos, baseando-se em literatura existente na área;
4. **Modelagem dos casos de projetos** – Definir como a informação sobre os projetos de software será representada no sistema, baseando-se nos critérios de similaridades e informações relevantes para o usuário;

5. **Desenvolvimento de protótipo** – Implementação de uma ferramenta para apoio na identificação de riscos baseada no modelo proposto;

1.3 Metodologia

Para alcançar o objetivo proposto foi selecionado um conjunto de ações divididas em quatro etapas distintas. Essas etapas são: Estudo da literatura, Modelagem, Implementação do protótipo e Análise dos resultados.

A primeira etapa divide-se em três ações:

- **Estudo de caracterização de projetos** – Em um projeto encontramos diversas características como tempo, custo, tamanho da equipe, etc. Porém é preciso identificar quais informações são relevantes para a taxonomia de projetos de software, e a partir dessa identificação construir um modelo de classificação. Desse modo apenas as informações necessárias serão armazenadas e posteriormente recuperadas.

- **Estudo de técnicas de RBC** – Sistemas RBC utilizam várias técnicas para comparar uma situação ou descrição de um problema (um caso) com uma base dados de casos conhecidos [3]. Portanto, é preciso identificar qual técnica melhor se aplica à solução proposta por este trabalho.

- **Estudo sobre identificação de riscos** – Este trabalho propõe uma abordagem para identificação de riscos de um projeto, porém é preciso conhecer outros aspectos de identificação de riscos já existentes, a fim de obter um melhor embasamento da solução proposta.

A segunda etapa é a modelagem do sistema de caracterização, armazenamento, comparação e recuperação das informações de projetos. A modelagem terá como apoio documentos de especificação do modelo de dados e método de comparação entre casos. Este modelo deve especificar todo o ciclo de identificação de riscos, como mostra a Figura 1.1. À medida que o sistema for utilizado novas informações são incorporadas. Pode-se dizer que riscos estão diretamente associados às organizações responsáveis pelo projeto. Cada organização tem diferentes qualidades e dificuldades, e estas podem ser fontes de riscos de um projeto, e.g., uma

organização que enfrenta problemas de transporte ou localização, onde seus equipamentos de hardware não são facilmente substituíveis. Um fato como esse muitas vezes é desconhecido ou esquecido por um gerente com pouca experiência dentro da organização. Contudo, com o ciclo de identificação proposto, o sistema será capaz de "aprender" sobre a organização onde atua, visto que está sendo alimentado de novas informações a cada novo projeto.

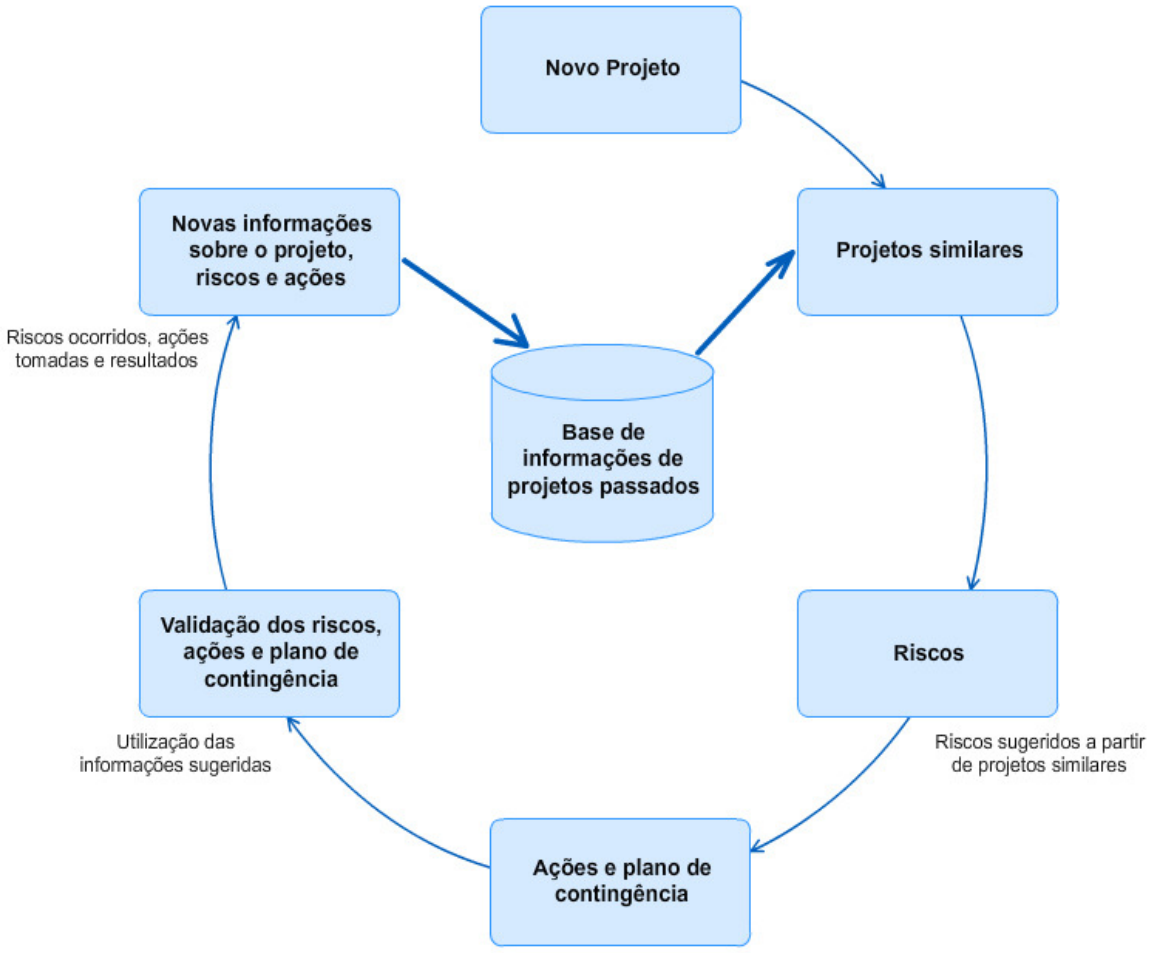

Figura 1.1 – Adaptação do ciclo RBC para identificação de riscos [4].

Na terceira etapa, será utilizado o modelo definido para a implementação de um protótipo utilizando banco de dados MySQL[1] e interface web PHP[2]. Este protótipo importará um documento HTML gerado pelo Microsoft Project, este documento é baseado em um mapa que

[1] MySQL, um banco de dados gratuito: www.mysql.com

[2] PHP: Hypertext Processor. Mais informações em www.php.net

define quais informações do projeto deverão ser exportadas. Depois disso será feita uma leitura e o processamento das informações contidas no mesmo.

Na última etapa os resultados da classificação feita pelo protótipo serão analisados. De acordo com estes resultados poderá ser necessário fazer ajustes ao modelo anteriormente definido. Esse processo poderá ser repetido até ser alcançado o desempenho esperado, ou seja, uma identificação eficiente de projetos de software similares, fornecendo informações relevantes para a identificação de riscos de projeto.

1.4 Estrutura do Documento

Após este capítulo introdutório e para um melhor entendimento das ações realizadas para alcance dos objetivos propostos, este documento é divido da seguinte forma:

Capítulo 2 – **Raciocínio Baseado em Casos** – este capítulo apresenta a técnica de Raciocínio Baseado em Casos, apresentando o estado da arte e mostrando os principais métodos utilizados para a construção de sistemas RBC.

Capítulo 3 – **Gerência de Riscos** – este capítulo tem a finalidade de apresentar a área de Gerência de Riscos de Projetos, mostrando sua importância em projetos de software.

Capítulo 4 – **Identificando Riscos de Projetos com base em RBC** – este capítulo apresenta, mostrando em detalhes, a proposta sugerida por este trabalho para construção de um sistema RBC para identificação de riscos de projetos de software com base em experiências anteriores.

Capítulo 5 – **Conclusão e Trabalhos Futuros** – este capítulo tem como objetivo apresentar resultados e considerações finais sobre o modelo proposto, além de apresentar sugestões de trabalhos futuros na área.

Capítulo 2 – Raciocínio Baseado em Casos (RBC)

Este capítulo tem como objetivo introduzir os conceitos de Raciocínio Baseado em Casos, mostrar suas principais características e dar uma visão do estado da arte para o leitor.

Raciocínio Baseado em Casos é uma técnica que busca uma solução para um problema novo baseando-se em experiências passadas. Vários sistemas baseados nesta técnica foram desenvolvidos nos mais diversos domínios de conhecimento, um exemplo é o sistema JULIA, utilizado no planejamento de refeições, utiliza não só experiência passada como também um conjunto restrições fornecidas pelo usuário (e.g. tipo de comida, valor máximo da refeição, número de pessoas, etc.) [4, 11]. Outro exemplo é o sistema CASEY, que usado na classificação de doenças cardíacas a partir da descrição de sintomas, fazendo uma comparação com sintomas similares identificados em outros pacientes [4, 11].

Na seção 2.1 desde capítulo, vamos apresentar a representação do conhecimento em um sistema RBC, suas técnicas e características. Em seguida, a seção 2.2, trata de similaridade, mostrando como podem ser feitas as comparações entre casos através de técnicas de cálculo de similaridade entre casos.

Um sistema RBC recupera um problema anterior que mais se aproxima do problema atual apresentado. As informações recuperadas geralmente trazem a forma como este problema anterior foi resolvido. Através dessa informação uma solução para o novo problema pode ser gerada adaptando-se ou não a solução utilizada no passado. Depois de aplicada a solução gerada, temos um novo caso que é armazenado no banco de dados do sistema, e passa a fazer parte dos casos disponíveis para uma nova busca.

2.1 Representação do Conhecimento

A representação do conhecimento é um aspecto essencial de um sistema RBC, onde são utilizadas unidades de conhecimento denominadas *casos*. Um *caso* é tipicamente composto pela descrição do problema e de sua respectiva solução. O problema descreve a situação, contexto ou características do caso. A solução pode ser a descrição de uma ação tomada ou qualquer informação útil ao usuário sobre o problema.

A descrição do problema precisa conter informações que devem dar suporte para a aplicação de uma regra de similaridade, ou seja, as informações devem ser suficientes para que seja possível julgar se um determinado caso passado é parecido com o caso atual. Num sistema de diagnóstico de problemas mecânicos em automóveis, por exemplo, informações relevantes poderiam ser aquelas que descrevem o estado em que o carro se encontra, e.g. o ar-condicionado não funciona, temperatura do motor elevada, rotação abaixo do normal, etc.

Neste trabalho os casos representam projetos de software, portanto ao invés de nos preocupar com o estado em que um problema ocorre, assim como no exemplo do parágrafo anterior, serão utilizadas características de projetos de software para que este seja comparado com projetos anteriores. A caracterização de projetos de software será discutida com detalhes no capítulo 4.

A solução de um caso varia de acordo com o domínio de conhecimento em que o sistema RBC é aplicado. De uma forma geral a solução de um caso contém informações necessárias para atingir um objetivo específico. No exemplo de diagnóstico de problemas em automóveis a solução é uma ação tomada pelo mecânico para resolver o problema, e.g. trocar a bomba de combustível, fazer alinhamento dos pneus, trocar o óleo de freio, etc. Além disso ainda pode ser adicionado à solução um *feedback* sobre o efeito causado por uma determinada ação, sendo este uma resposta do usuário ao sistema em que ele descreve o que ocorreu após ter agido no problema em questão, e.g. rotação foi estabilizada, ar-condicionado parou de funcionar, o carro não apresentou nenhuma alteração.

Contudo, neste trabalho propomos que a solução para um caso representa as informações sobre os riscos de um projeto de software, contendo os riscos identificados para um projeto anterior, as estratégias definidas para seu tratamento e as respectivas respostas (quais deles tiveram impacto no projeto e os planos de contingência adotados).

A forma de representar o conhecimento no sistema RBC também é uma característica importante, vários tipos de linguagem ou modelos de representação podem ser utilizados. Na seção 2.1.1 serão abordadas algumas técnicas de representação mais comumente utilizadas por sistemas RBC.

2.1.1 Representação Atributo-Valor

Em uma representação atributo-valor temos um conjunto de tuplas <nome do atributo, valor> utilizadas para representar informações de um determinado domínio, e.g. <peso, 5Kg>. Esta é uma das formas mais simples de representação de casos.

Em geral, nesse tipo de representação, cada atributo é associado a um tipo (domínio). Os tipos básicos mais comumente utilizados são: número, booleano, data, símbolo e *string*. Cada atributo tem uma faixa de valores, no caso de um atributo numérico, e.g. quantidade, os valores poderiam variar de zero a mil.

Símbolos são atributos cujo valor faz parte de um conjunto pré-determinado de valores simbólicos, e.g. num sistema que guarda informações sobre carros teríamos um atributo <marca do carro, valor>, onde o valor faz parte de um conjunto de marcas de automóveis mais comuns no mercado. Os atributos do tipo símbolo podem ser agrupados de três formas: ordenados, não-ordenados ou hierárquica (taxonomia), detalhadas a seguir.

- **Símbolos ordenados –** são valores do tipo símbolo que se apresentam em uma ordem pré-definida, de forma que os valores mais semelhantes fiquem mais próximos, e.g. um conjunto de símbolos que representam a qualidade de um determinado produto:
    - {péssimo, ruim, satisfatório, bom, muito bom, excelente}, neste caso os valores possuem uma ordem de precedência que indica que símbolo “bom” é mais parecido com símbolo “satisfatório” do que com símbolo “péssimo”.
- **Símbolos não-ordenados –** são valores que não usam nenhuma forma de precedência, portanto podemos apenas comparar se dois atributos tem o mesmo valor ou não. No exemplo de informações sobre automóveis o atributo marca seria do tipo símbolo não-ordenado.
- **Símbolos ordenados por taxonomia –** têm seus valores ordenados através da relação entre os mesmos. Esta relação de similaridade é bem mais complexa do que uma simples ordem de precedência citada anteriormente, desta forma é possível ter um “mapa” para

calcular a “distância” entre atributos, sendo muito útil para domínios de conhecimento mais complexos.

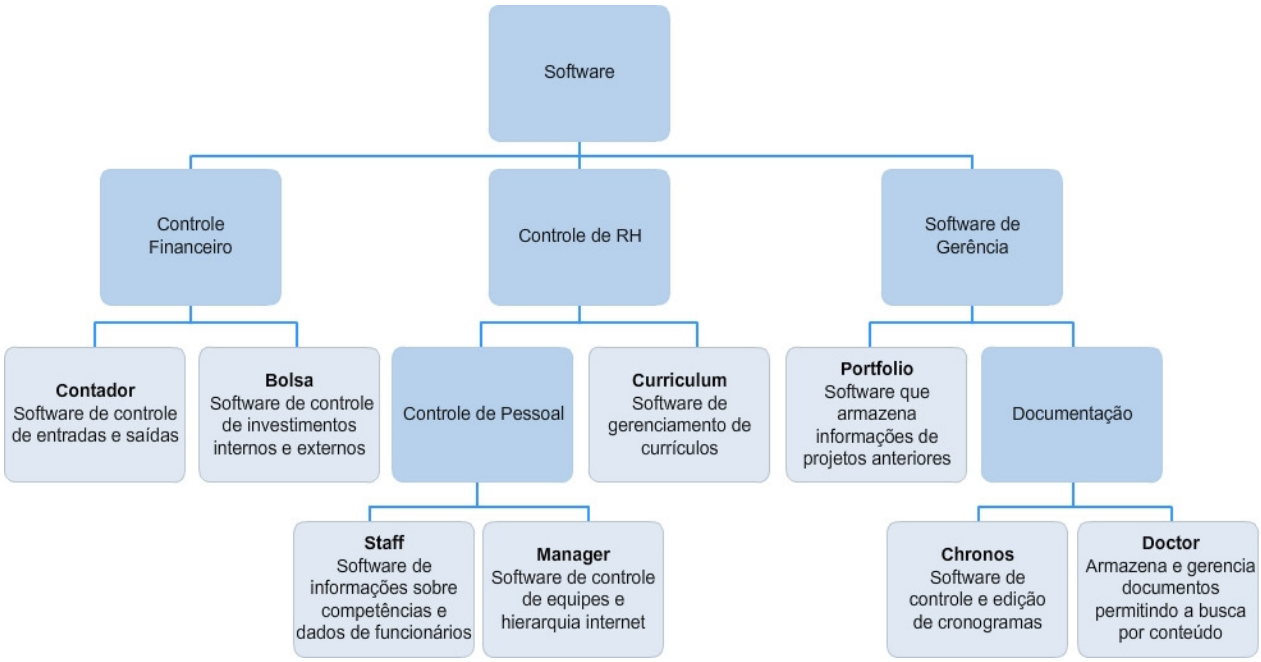

Figura 2.1 – Exemplo de Taxonomia para Tipos de Software

No exemplo da figura 2.1 podemos observar que o programa *Staff* está mais próximo do programa *Curriculum* do que do programa *Doctor*, portanto a similaridade entre *Staff* e *Curriculum* é maior.

A representação por atributo-valor não apresenta nenhuma informação relacional entre os casos. Normalmente ela é utilizada em sistemas de diagnóstico que lidam com grandes quantidades de casos.

2.1.2 Representação Orientada a Objetos

Na representação orientada a objetos, de forma análoga a linguagens de programação, os casos são divididos em classes, onde cada caso é uma instância de uma classe que contém atributos, estes por sua vez também podem ser objetos, ou seja, instâncias de outras classes. Uma classe pode ser vista como a definição de um tipo. Casos que fazem parte de uma mesma classe podem ser considerados similares.

Representações orientadas a objetos permitem a representação de relações em diversos níveis. Os atributos de uma classe podem referenciar outros objetos, permitindo a modelagem de domínios de aplicação mais complexos.

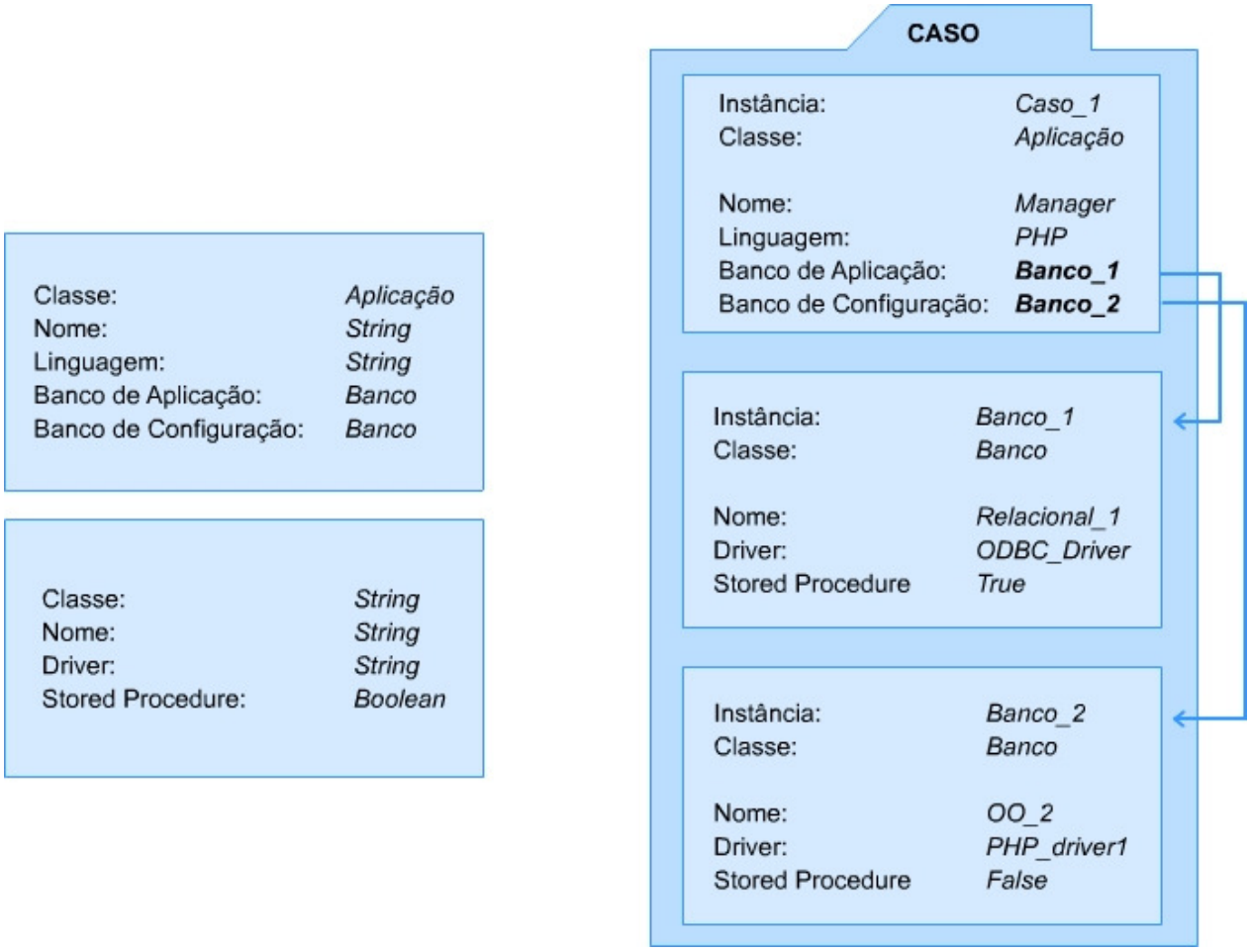

Figura 2.2 – Adaptação da Representação Orientada a Objetos [4].

No entanto, este tipo de representação torna o cálculo de similaridade bem mais complexo se comparado com um cálculo utilizado em uma representação atributo-valor. Na Figura 2.2 podemos ver a relação entre as classes Aplicação e Banco. Neste caso uma técnica de cálculo de similaridade entre objetos é utilizada.

A representação orientada a objetos permite que informações sobre herança sejam armazenadas e utilizadas no cálculo de similaridade. Além disso, os atributos podem conter informações mais detalhadas, podendo até conter sub-atributos ao invés de um único valor.

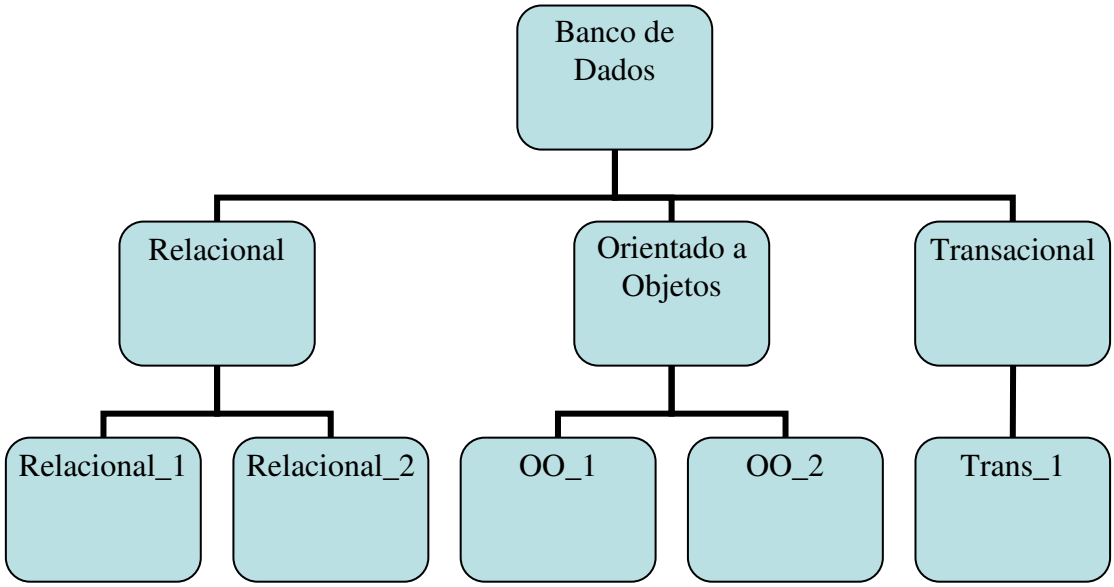

Figura 2.3 – Exemplo de hierarquia de objetos de banco de dados.

Ao utilizar uma representação orientada a objetos a similaridade é, analogamente, calculada entre objetos, levando em consideração as relações hierárquicas dos mesmos. A similaridade entre objetos é inicialmente calculada em nível de atributos. No exemplo da Figura 2.2, o atributo *Banco* é também um objeto, portanto para calcular a similaridade entre os atributos *Banco de Aplicação* e *Banco de Configuração* é preciso levar em consideração as relações representadas na Figura 2.3.

Portanto se tivermos, por exemplo, um caso A, onde o *Banco de Aplicação* utilizado é *Relacional_2* e um *Banco de Configuração* utilizado é *OO_1*, e um caso B, onde tanto o banco de aplicação quanto o de configuração utilizam *Trasn_1*, então poderemos dizer que o caso representado na Figura 2.2 é mais similar ao caso A do que ao caso B.

2.1.3 Representação Através de Estruturas

Outra abordagem utilizada em sistemas RBC é a representação por redes semânticas. Uma rede semântica é um tipo de grafo com nodos e arestas dirigidas, que representam relacionamento entre os nodos. Um tipo de rede semântica, a RRC (Rede de Recuperação de Casos), foi desenvolvida como uma técnica mais flexível, capaz de trabalhar com dados ambíguos, além de manipular bases de casos maiores de modo eficiente.

Além das redes semânticas, também são utilizadas estruturas em árvore para representação de casos em sistemas RBC. A principal estrutura utilizada são as *árvores k-d* [4],

árvores de busca binária onde cada nodo representa um subconjunto dos casos da base de casos, nas extremidades (folhas) encontram-se os casos e a raiz, representa todos os casos.

2.2 Similaridade

A definição de similaridade em um sistema RBC é talvez o aspecto mais importante a ser levado em consideração na hora construir o sistema, é também um conceito que pode variar de acordo com a sua aplicação e o domínio de conhecimento em que se trabalha.

Na prática, ao se recuperar um caso em um sistema RBC o objetivo é trazer um caso que seja útil para resolução de um problema. O conceito de utilidade ou relevância pode ser muito difícil de ser determinado, portanto na construção de sistemas RBC, de uma forma geral, é possível ter como base a seguinte hipótese: "Problemas similares possuem soluções semelhantes" [4].

Neste trabalho o objetivo do sistema é trazer um caso (projeto de software) cujos riscos podem ser os mesmos dado um novo projeto, portanto a similaridade entre os casos está diretamente ligada aos riscos dos mesmos. A partir dessa idéia podemos elaborar a seguinte pergunta: **Supondo que dois projetos de software têm os mesmos riscos, como estes projetos podem ser caracterizados?**

Esta questão será discutida em detalhes no capítulo 4 deste trabalho, onde vamos examinar os detalhes de cada atributo escolhido para caracterizar projetos de software.

Ao se definir o critério de similaridade de um sistema RBC é preciso levar em consideração diversos aspectos. Abordaremos a seguir alguns deles.

- **Indexação –** Os atributos usados para o cálculo de similaridade entre dois casos são chamados de índices. Os índices representam as características que identificam o caso como sendo relevante ao problema dado.
- **Propriedades da similaridade –** O conceito de similaridade pode variar de acordo com o domínio de aplicação e os objetivos específicos de um sistema RBC. De acordo com a similaridade definida esta pode apresentar, ou não, as seguintes propriedades:
    - **Reflexividade –** Um caso é parecido com si mesmo.
    - **Simetria –** Se um caso X é parecido com um caso Y, então Y é parecido com X. Apesar de parecer uma afirmação óbvia, este conceito não necessariamente prevalece [4]. A idéia de que deve existir simetria em sistemas em sistemas RBC é questionada por alguns autores. Como exemplo tem-se a afirmação: "Ele se

parece com o presidente", neste caso podemos considerar que a similaridade de um elemento de uma categoria (o sujeito) em relação ao protótipo (o presidente) é considerada maior do que a similaridade do protótipo a um determinado elemento da categoria ("O Presidente se parece com este cidadão?") [4].

- **Transitividade –** Se um caso X é parecido com Y e este caso parecido com Z, então X é parecido com Z. Esta propriedade é geralmente rejeitada na criação de sistemas RBC. É fácil entender através de exemplos o porque isto acontece: "Se um triângulo vermelho é parecido com um quadrado vermelho, e um quadrado vermelho é parecido com um quadrado azul, então um triângulo vermelho é parecido com um quadrado azul" [4].

2.2.1 Medidas de Similaridade

Existem diversas técnicas para medir a similaridade entre dois casos, a escolha de uma delas fica a critério do projetista do sistema. Para uma escolha adequada de qual técnica deve ser usada é preciso analisar os aspectos do domínio de aplicação, além disso, é preciso levar em conta a técnica de representação utilizada para representar os problemas passados na base de casos.

Podemos dividir as medidas de similaridade em dois tipos:

- **Similaridade global** – medida utilizada na comparação entre casos levando em consideração todos os seus índices.
- **Similaridade local** – medida utilizada na comparação entre índices, onde cada índice de um caso é comparado individualmente com os índices do caso atual.

A seguir vamos abordar alguns exemplos de técnicas muito utilizados para calcular a similaridade, tanto global como localmente.

- **Nearest neighbour**[1] **–** Esta é uma medida de similaridade global bastante simples e muito utilizada. Nesta abordagem os casos são interpretados como pontos em um plano ou espaço dimensional. Como exemplo podemos imaginar um plano onde suas arestas representam valores para dois atributos, neste plano um caso é representado por um par de índices (x,y), que funcionam como coordenadas do plano, a partir disso é calculada uma distância entre os casos somando a diferença no eixo x à diferença no eixo y, o resultado é o valor da distância entre os casos. Ainda é possível utilizar pesos para cada atributo, alterando o cálculo da similaridade.

[1] Vizinho-mais-próximo. Neste trabalho será utilizado o termo original.

- **Uso de pesos –** Pesos podem ser utilizados no cálculo da distância, no exemplo citado onde temos um par de eixos (x,y) pode-se utilizar pesos W_x e W_y para os atributos x e y, assim temos:
  - $D_1 = (x_1 \times W_X) + (Y_1 \times W_Y)$
  - $D_2 = (x_2 \times W_X) + (Y_2 \times W_Y)$

- **Similaridade entre objetos** – Esta é uma medida de similaridade global que, de modo diferente do *nearest neighbour*, leva em consideração as informações de hierarquia de classes, modelada de acordo com o domínio de aplicação, previamente utilizada na representação dos casos. Na hierarquia de classes objetos mais próximos têm similaridade maior, com isso podemos inferir que a estas informações se assemelham com uma representação taxonômica. A comparação de dois objetos pode ser feita através das seguintes abordagens:
  - **Similaridade intraclasse** – Comparação feita entre objetos de mesma classe, utiliza apenas os atributos do objeto.
  - **Similaridade interclasse** – Comparação feita entre objetos de classes diferentes que herdam de uma mesma classe pai. Muito similar a comparação entre símbolos taxonômicos.
- **Medidas de similaridade local** – Em um mesmo caso podemos encontrar índices de tipos diferentes, para cada tipo é possível utilizar técnicas diferentes para compará-los. As medidas de similaridade local dependem diretamente do domínio de aplicação utilizado.
  - **Número** – Atributos representados por número podem ser comparados diretamente, por exemplo, atributos com valores 100 e 200, neste caso é possível calcular diretamente a diferença entre os valores, que é igual a 100, e assim determinar a sua similaridade. Porém podemos observar em outro exemplo que valores como 10100 e 10200 têm a mesma similaridade, portanto a proporcionalidade foi descartada, todavia podemos usar uma escala logarítmica, desta forma levaremos em conta também as proporções dos valores, usando um logaritmo de base 10 temos que a similaridade entre os valores 10 e 20 é a mesma que os valores 1000 e 2000.
  - **Símbolo ordenado** – Ao utilizar símbolos ordenados podemos atribuir valores numéricos aos mesmos e assim utilizar as mesmas medidas utilizadas nos tipos numéricos, exemplo:

- {ótimo → 1, muito bom → 2, bom → 3, ruim → 4, inaceitável → 9}

- **Símbolo não ordenado** – Símbolos não ordenados podem ter seus valores de similaridade representados em uma tabela, esta é construída de acordo com o contexto de utilização do sistema. Na Tabela 2.1 vemos um exemplo de valores de similaridade para tipos de projetos de software.

Tabela 2.1 – Valores de similaridade entre projetos de software.

v_i / v_k	Web	Aplicação	Sistema Embarcado
Web	1	0.6	0.1
Aplicação		1	0.3
Sistema Embarcado			1

- **Símbolos taxonômicos** – Símbolos ordenados taxonomicamente são dispostos em uma árvore onde cada nodo carrega um valor numérico que simboliza a similaridade entre seus nodos filhos, quanto mais próximo das extremidades maiores os valores. No exemplo da Figura 2.4 podemos observar que similaridade entre *Chronos* e *Doctor* é de 0.7 pais o nodo pai mais próximo é **Documentação**, já entre *Chronos* e *Portfolio* é de 0.2, neste caso o nodo pai mais próximo comum aos dois é **Software de Gerência**.

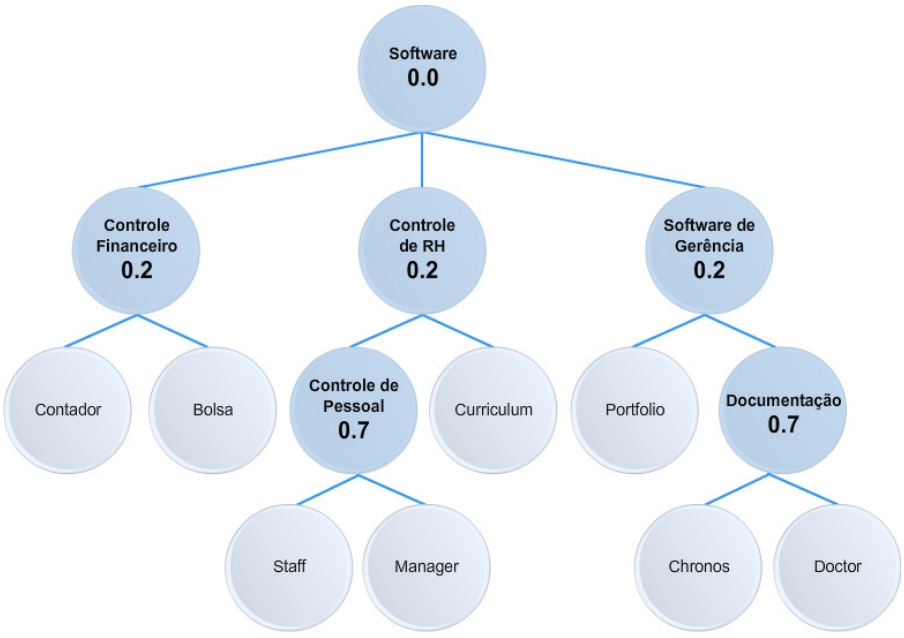

Figura 2.4 – Exemplo de Taxonomia de Software.

2.3 Recuperação

Para atingir o objetivo final de um sistema RBC, que é auxiliar na resolução de problemas baseando-se em experiências passadas, é preciso recuperar da base de dados os casos que podem ser relevantes para o problema atual. Depois de definir como as informações são representadas na base de casos (seção 2.1), e de como será feito o cálculo de similaridade entre dois casos (seção 2.2), é hora de definir um processo para a recuperação dos casos considerados relevantes ao problema atual.

Nesta seção serão mostrados os princípios gerais sobre recuperação de casos em um sistema RBC. O processo de recuperação pode ser divido em três subtarefas: *assessoramento da situação*, *casamento*, *seleção* e *recuperação seqüencial*.

O assessoramento da situação é uma técnica onde é formulada uma consulta baseada em um conjunto de informações sobre a situação ou problema atual, na maioria dos casos essas informações são fornecidas ao sistema pelo próprio usuário. A dificuldade desta técnica reside no fato de que as informações do mundo real geralmente são muito complexas, tornando difícil identificação de quais informações são essenciais sobre o problema.

Durante o projeto do sistema é feita uma indexação para eleger quais atributos identificam o caso (seção 2.2), estes são usados como descritores de entrada, a definição de quais descritores são relevantes para encontrar a solução do problema é uma tarefa extremamente difícil. Pode-se dizer que o objetivo desta subtarefa é identificar o contexto em que o problema se apresenta e filtrar ruídos e descritores não relacionados com o problema.

O casamento é uma técnica que utiliza informações sobre o problema atual como base para criação de uma consulta e depois faz uma comparação com os casos da base de casos. Porém ao fazer a combinação de consulta e comparação esta pode ter um custo computacional relativamente elevado. As tarefas de casamento e assessoramento podem ser executadas de forma alternada em alguns sistemas.

A seleção é a tarefa que analisa um conjunto de casos similares ao problema, e escolhe a melhor opção dentre os casos do conjunto, isto pode ser feito durante a execução da tarefa de casamento, mas geralmente esta tarefa retorna um conjunto de casos onde a tarefa de seleção é executada. Esta escolha é feita através da aplicação do cálculo de similaridade feito de forma detalhada usando o conhecimento sobre o domínio de aplicação. Também é possível requisitar ao usuário que confirme qual escolha é mais adequada ou ainda que forneça alguma informação

adicional afim de escolher dentre um subconjunto de casos qual é a melhor solução para o problema.

A *recuperação seqüencial* é uma técnica simples, onde o cálculo de similaridade é aplicado a todos os casos da base de dados seqüencialmente, em seguida os casos são ordenados de acordo com a similaridade em relação ao problema, depois disso são eleitos os *n* casos mais similares. Esta não só é uma técnica simples e oferece algumas vantagens:

- O processo de similaridade é completo, pois temos a garantia de que todos os casos serão comparados um a um com o problema, não havendo a possibilidade de um caso que contenha a solução para o problema não ser comparado ao problema.
- A consulta na base de casos independe da medida da similaridade, podendo desse modo ser combinada com diferentes técnicas em um mesmo sistema RBC.
- Fácil implementação.

2.4 Resumo do Capítulo

Neste capítulo apresentamos a idéia por trás de um sistema RBC, abordando seus objetivos e como esta técnica pode ser aplicada em diversos domínios de conhecimento podendo ajudar na resolução de vários tipos de problemas do mundo real.

Além de mostrar onde sistemas RBC podem ser usados, foram abordadas algumas das principais técnicas de representação de informações sobre problemas resolvidos no passado através de unidades de conhecimento chamadas de casos.

Embora a metodologia de representação seja de grande importância para o sistema, é essencial que mesma dê suporte ao cálculo de similaridade. Abordamos também, neste capítulo, os principais métodos para calcular a similaridade entre casos. Em geral estes métodos são divididos em similaridade global, que é a comparação entre casos, e a similaridade local, que faz uma comparação entre atributos.

Depois de definir como representar e calcular a similaridade entre casos, é preciso definir um processo de recuperação para encontrar um ou mais casos relevantes para o problema. Neste capítulo vimos que este processo é dividido em sub-tarefas e quais são suas características.

Capítulo 3 – Gerência de Riscos

Neste capítulo vamos abordar a disciplina de Gerência de Riscos que, no contexto deste trabalho, tem o importante papel de relacionar o modelo proposto com o objetivo final do sistema RBC, que é o de auxiliar o gerente de projetos a fazer a identificação dos riscos de um novo projeto. Além disso, o estudo da Gerência de Riscos faz parte do domínio de conhecimento do sistema RBC proposto neste trabalho e ainda nos ajuda a mostrar a sua relevância.

Na seção 3.2 encontra-se uma descrição detalhada do processo de Gerência de Riscos conforme o Guia PMBOK. Na seção 3.3 são apresentados diversos aspectos da identificação de riscos, contextualizando o trabalho desenvolvido neste documento.

3.1 Importância da Gerência de Riscos

A construção de softwares hoje em dia é um grande desafio para a maioria das organizações envolvidas nesta atividade. Com um mercado cada vez mais exigente, decisões rápidas, melhor alocação de recursos e metas bem traçadas tornam-se, cada vez mais, pré-requisitos para o sucesso de projetos de software [5].

Atualmente é comum encontrar organizações da indústria de software que enfrentam problemas relacionados a qualidade, custo e prazo. O sucesso de projetos de software está intimamente ligado ao sucesso da própria organização que os gerem, portanto boas práticas de gestão de software podem ser encaradas como boas práticas de gestão de negócio [6].

É importante entender a diferença entre a identificação de riscos e a gerência dos mesmos. O processo de identificação de riscos além de ser um pré-requisito necessário para gerenciar riscos, também dá suporte à avaliação dos riscos de um projeto. Neste caso, um risco de alta probabilidade e impacto pode não ser aceitável, fazendo que o projeto seja cancelado ou re-estruturado a fim de evitar maiores danos às organizações envolvidas no mesmo.

O gerenciamento de riscos, por sua vez, é responsável por um monitoramento dos riscos durante o desenvolvimento do projeto, desta forma uma resposta adequada ao risco pode ser utilizada, e os impactos negativos causados pelo menos podem ser minimizados.

A Gerência de Riscos é tradicionalmente vista como sendo uma parte da Gerência de Projetos por diversos autores, porém a Gerência de Riscos também pode ser encarada como uma disciplina independente, ou ainda, pode-se dizer que a razão da Gerência de Projetos é a própria Gerência de Riscos.

3.2 Gerência de Riscos no Guia PMBOK

De acordo com o Guia PMBOK o Processo de Gerenciamento de Riscos é dividido nas seguintes fases:

- **Planejamento do gerenciamento de riscos –** Nesta fase são definidas as atividades do Processo de Gerenciamento de Riscos, assim pode-se garantir que o nível, tipo e visibilidade do processo sejam adequados. O plano de Gerência de Riscos inclui, dentre outros, os seguintes itens:
    - **Metodologia –** Onde são definidas técnicas e ferramentas que podem ser utilizadas no processo.
    - **Funções e responsabilidades –** Uma descrição de como será a participação da equipe nas atividades do processo.
    - **Categoria dos riscos –** Fornece uma classificação dos riscos para auxiliar na identificação dos mesmos.
    - **Definição de probabilidades e impacto de riscos –** Gera uma escala de probabilidade e impactos, através da qual é possível analisar quais riscos podem representar uma maior ameaça ao projeto.
    - **Matriz de probabilidade e impacto –** Um cruzamento das informações de probabilidade de ocorrência de um risco com o seu nível de impacto no projeto, servindo de base para a definição de prioridades para o tratamento de riscos.
- **Identificação de Riscos –** Cria uma documentação preliminar que lista os possíveis riscos de projeto e detalha suas características. Na seção 3.3 vamos analisar em detalhes a identificação de riscos de projetos.
- **Análise qualitativa de riscos –** Avalia a prioridade dos riscos anteriormente identificados, baseando-se na probabilidade de ocorrência e impacto sobre o projeto.

- **Análise quantitativa de riscos –** Atribui uma classificação numérica aos riscos priorizados na Análise qualitativa de riscos, alguns gerentes de riscos mais experientes preferem fazer esta análise logo após a identificação de riscos de projetos.
- **Planejamento de resposta a riscos –** Diversas estratégias podem ser utilizadas como resposta a riscos, porém cada risco deve ter um plano ou conjunto de estratégias de maior eficácia.
- **Monitoramento e controle de riscos –** Nesta fase os ricos são monitorados, isto pode ser feito através de diversas atividades como Reavaliação dos Riscos, Auditoria de riscos, Análise das tendências da variação, Medição do desempenho técnico, Análise das reservas e reuniões de andamento.

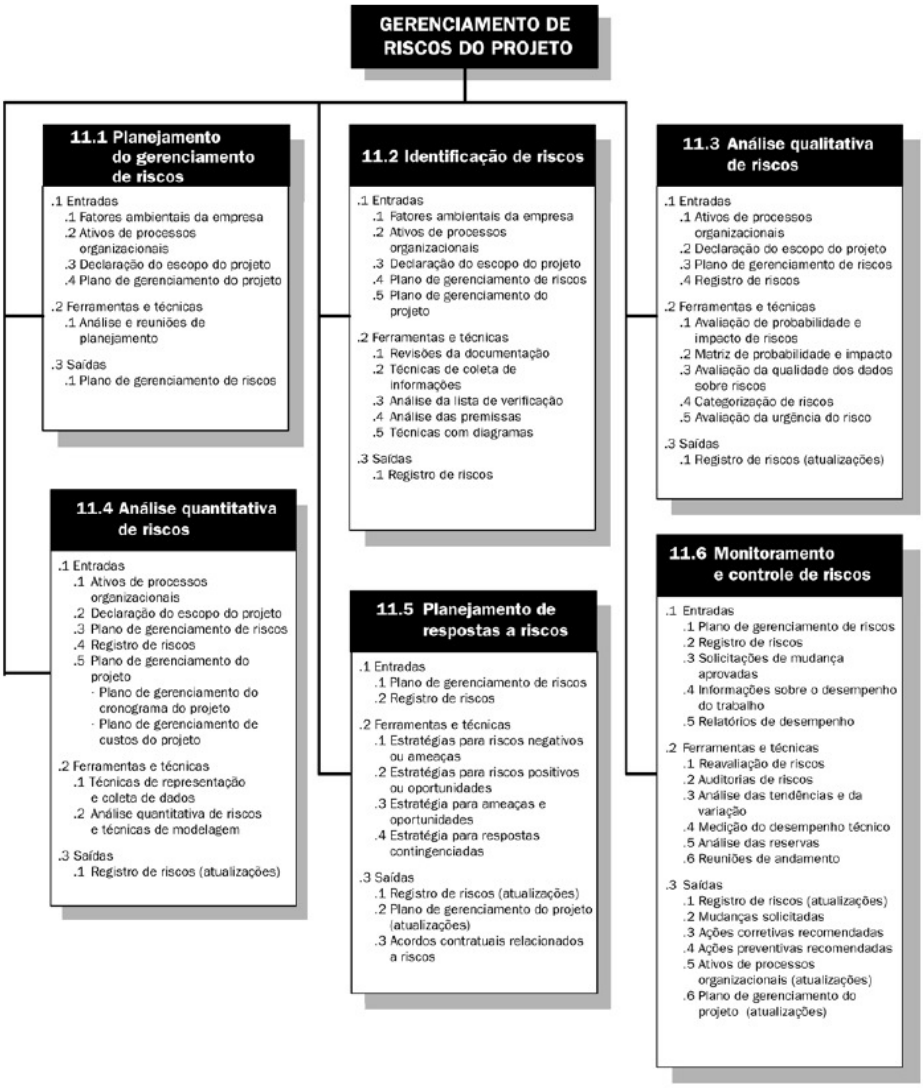

Figura 3.1 – Visão geral do Gerenciamento de Riscos de Projeto [1].

Na Figura 3.1 podemos observar uma estrutura analítica que descreve todo o Processo de Gerenciamento de Riscos sugerido pelo Guia PMBOK, onde as fases do processo recebem entradas e geram saídas. Na fase de *Identificação de riscos* podemos observar que a saída gerada é um *Registro de riscos*, sendo este re-utilizado como uma entrada nas fases de *Planejamento de resposta a riscos* e *Monitoramento e controle de riscos*.

Portanto torna-se evidente a importância da identificação de riscos dentro do Processo Gerência de Riscos, sendo, desta forma, uma motivação para o desenvolvimento deste trabalho.

3.3 Identificação de Riscos

A identificação de riscos é uma atividade comum a diversas abordagens de Gerência de Riscos, por exemplo: ISO-9000-3, ISO-12207 / ISO-15504, CMM[1], CMMI[2] [6]. Este trabalho tem como principal objeto a definição de um modelo para identificação automática de riscos, podendo ser utilizado em conjunto com qualquer um dos modelos citados anteriormente.

Um processo de identificação de riscos baseado em taxonomia proposto pelo SEI (*Software Engineering Institute*) [7] classifica riscos de projetos em três classes:

- Conhecidos – Aqueles que alguma pessoa da equipe conhece ou está ciente.
- Desconhecidos – São os riscos que podem vir à tona (ser identificados) caso seja dada uma oportunidade à equipe.
- Irreconhecíveis – Riscos que a principio ninguém pode prever.

Além disso, este método baseia-se nas seguintes suposições:

- Riscos geralmente são conhecidos pela equipe técnica, mas raramente são comunicados ao gerente.
- Um método de identificação de riscos estruturado e reusável é necessário para uma Gerência de Riscos consistente.
- O processo de identificação de riscos deve criar e manter um ambiente imparcial para que diferentes pontos de vista sejam ouvidos.
- Nenhum julgamento generalista pode ser feito sobre o sucesso ou falha de um projeto baseado apenas no número ou natureza dos riscos.

---

[1] CMM (*Capability Maturity Model*) – Mais informações em www.sei.cmu.edu/cmm/.

[2] CMMI (*Capability Maturity Model Integration*) – Mais informações em www.sei.cmu.edu/cmmi/.

O processo de identificação de riscos tem como base uma taxonomia de processo de desenvolvimento de software, que organiza as atividades do processo de desenvolvimento de software, servindo de base para organizar e estudar a amplitude dos riscos de projeto.

A taxonomia de processo de desenvolvimento de software sugerida pelo SEI, mostrada na Figura A.1, é usada como base para o TBQ (*Taxonomy-Based Questionnaire*[1]), que consiste de um conjunto de perguntas em nível de atributos juntamente com sugestões. O objetivo do TBQ é identificar áreas problemáticas, preocupantes ou que contenham riscos.

O processo de identificação de riscos trata de uma série de entrevistas com grupos da equipe selecionada para o projeto. Cada entrevista se dá em dois passos:

- Pergunta e Resposta: O TBQ e questões sensíveis ao contexto do projeto são usados para identificar dificuldades, preocupações e riscos que podem causar o insucesso do projeto.
- Esclarecimento dos problemas: Neste passo o risco é classificado dentro de uma taxonomia através de um consenso dos participantes. Riscos equivalentes são transformados em um único risco.

3.3.1 Técnicas para Identificação de Riscos

Vários métodos para identificação de riscos são encontrados na literatura de Gerência de Riscos. Os métodos (1) revisão da documentação, (2) técnica Delphi, (3) análise de lista de verificação e (4) análise das premissas são alguns dos métodos referenciados pelo Guia PMBOK.

- **Revisão da documentação** – Uma revisão na documentação do projeto pode ser realizada, desta forma é possível identificar problemas de inconsistência e qualidade nos planos de projeto, estes problemas podem, por sua vez, ser indicadores de risco de projeto.
- **Técnica Delphi** – O objetivo dessa técnica é obter um consenso entre especialistas. Um questionário é usado para que os especialistas escrevam suas idéias, anonimamente, sobre os riscos mais importantes do projeto. Estas respostas são redistribuídas entre o grupo, comentários são adicionados caso desejado. Desta forma um consenso imparcial pode ser atingido depois de algumas rodadas.
- **Análise de lista de verificação** – Listas de verificação de identificação de riscos podem ser criadas com base em informações de projetos passados e em experiência adquirida. Ao final do projeto a lista é revisada. Desta forma estas informações podem ser utilizadas em

[1] Questionário Baseado em Taxonomia – Neste trabalho é usado o termo original.

projetos futuros. O modelo de identificação de identificação de riscos proposto neste trabalho baseia-se neste método.

- **Análise das premissas** – As hipóteses e premissas tomadas como base para o projeto são analisadas e validadas ao longo de seu desenvolvimento. Desta forma pode-se prevenir que o projeto baseie-se em premissas irreais (imprecisas, inconsistentes, incompletas) [8].

3.4 Resumo do Capítulo

Este capítulo apresentou uma visão de Gerência de Riscos e sua importância no gerenciamento da construção de softwares a fim de evitar o insucesso dos mesmos. Uma atividade importante e comum à maioria dos processos de Gerência de Riscos existentes é a identificação dos riscos.

Foram apresentados também alguns métodos e técnicas de identificação de riscos como, por exemplo, o baseado em taxonomia de atividades de desenvolvimento de software desenvolvida pelo SEI. O referido método utiliza um questionário e entrevista com grupos da equipe de projeto para identificar quais processos do desenvolvimento apresentam riscos em potencial.

Capítulo 4 – Identificando Riscos de Projetos com base em RBC

Na criação de um sistema RBC é preciso definir o modelo de representação de conhecimento e cálculo de similaridade baseado em conhecimento empírico (como o próprio conceito de similaridade) e no conhecimento do que será modelado, neste trabalho nosso domínio de aplicação são os projetos de software. Neste capítulo são apresentados detalhes da modelagem proposta que tem o objetivo de definir um sistema de identificação de riscos de projetos de software utilizando experiências passadas.

Na seção 4.1 definiremos a representação dos valores de atributos escolhidos para representar projetos de software na forma de casos. A definição do cálculo de similaridade é apresentada na seção 4.2 deste capítulo. Os pesos de cada atributo serão discutidos na seção 4.3 e utilizados para encontrar qual dos casos armazenados na base de casos é mais parecido com o caso atual. Na seção 4.4 apresentamos com detalhe o cálculo de similaridade local e global utilizados no modelo proposto. Na seção 4.5 um modelo de protótipo é descrito para implementação do modelo proposto neste trabalho.

Para construir uma base de informações sobre projetos de software capaz de prover informações para um sistema RBC é preciso definir como o sistema trabalha com as informações fornecidas a ele. Isto é feito em uma série de etapas:

- Representação dos casos (projetos de software) no sistema;
- Caracterização dos casos;
- Definição do cálculo de similaridade.

A caracterização de projetos de software, no contexto deste trabalho, não é uma tarefa simples, pois estaremos lidando com diversos conceitos abstratos, e.g. o conceito de similaridade (entre projetos de software) e o levantamento de riscos de projeto, que também é uma disciplina que envolve formas de conhecimento empíricas como a experiência e a intuição do gerente de projetos.

Portanto, a modelagem discutida a seguir não deve ser encarada como um modelo rígido, e sim uma possível solução para a criação de um sistema de suporte a decisão na área de gerência de riscos de projeto. No capítulo 5, veremos sugestões para melhorias e adaptação do modelo proposto.

4.1 Representação

Como vimos no capítulo 3, existem diversas formas de representar o conhecimento em forma de casos. No entanto, a escolha de um modelo para representar projetos de software não é uma tarefa trivial. Para fazer uma escolha adequada é preciso analisar como estes modelos de representação podem ser usados no domínio de conhecimento que abordamos neste trabalho.

As representações que utilizam relacionamento taxonômico entre os casos, representação orientada a objetos e árvores estruturadas, podem ser de grande ajuda no cálculo da similaridade, pois trazem informações sobre as relações que existem entre os casos armazenados na base de casos (seções 2.1.2 e 2.1.3). Porém, a utilização de uma dessas abordagens necessita de um profundo conhecimento sobre o domínio da aplicação em questão. Caso alguma relação seja representada incorretamente o desempenho de todo o sistema será afetado, visto que tanto a recuperação quanto o cálculo de similaridade serão baseados em informações incorretas.

Podemos imaginar que numa representação orientada a objetos para projetos de software, estes projetos poderiam ser classificados pelo tipo de produto desenvolvido, e.g. uma aplicação financeira ou um sistema operacional para um dispositivo móvel. Porém esta abordagem pode trazer sérias desvantagens. Uma delas é que nem sempre projetos de softwares que constroem produtos diferentes são necessariamente projetos diferentes entre si, estes podem ter muitas características em comum, sendo portanto considerados projetos parecidos, encontrar projetos parecidos é suficiente para o sistema proposto neste trabalho que parte da suposição que projetos de softwares similares podem apresentar riscos semelhantes.

Portanto, o método escolhido para a modelagem proposta é o método de representação por atributo-valor. Apesar de este método ser usualmente utilizado em sistemas de diagnóstico com grande base de casos, ele traz vantagens de implementação simples, tanto da aplicação como do armazenamento dos dados. Além disso, os atributos utilizados podem usar diversas representações, inclusive representações mais complexas como símbolos ordenados por taxonomia (seção 2.1.1), tornando possível a construção de um cálculo de similaridade mais complexo.

Além da utilização de uma representação de atributo-valor, a recuperação dos dados feita no sistema proposto não vai utilizar nenhuma restrição, ou seja, o caso atual será sempre comparado com todos os casos da base de casos. Esta abordagem pode trazer uma significativa perda de desempenho ao sistema, porém a massa de dados do sistema será pequena na maior parte do tempo, já que a quantidade de projetos de software desenvolvidos em uma organização geralmente não é tão grande.

4.2 Definição da Caracterização

O principal objetivo da caracterização de projetos de software neste trabalho é a identificação de índices capazes de fornecer informações suficientes para o cálculo de similaridade entre os casos. No desenvolvimento do Modelo de Adaptação de Processo de Software (MAPS) [9], foi feito um estudo de caracterização de projetos de software. Neste trabalho vamos eleger algumas dessas características consideradas relevantes para o domínio de aplicação abordado:

- Tamanho da equipe
- Distribuição geográfica
- Experiência da equipe de desenvolvimento
- Tamanho do projeto

Outras características sugeridas no MAPS não foram utilizadas, pois estão mais relacionadas ao processo de construção de software do que com o projeto de software. As características não aproveitadas foram:

- Padrões adotados
- Exigências Contratuais
- Ferramentas Disponíveis
- Criticidade do software
- Orçamento
- Cronograma

O cronograma de um projeto de software é capaz de caracterizar aspectos do projeto como gerência e prazos além de ter relação direta com riscos relacionados a cumprimento dos prazo e custo do projeto. Porém, a manipulação de todas as informações contidas no cronograma apresenta um nível alto de complexidade para definição de valores e cálculo de similaridade, podendo assim dificultar a recuperação de casos no sistema se não apresentar uma modelagem adequada para uma mesma organização.

Além dessas características, mais outras duas também foram adicionadas:

- Tipo de Projeto
- Plataforma Tecnológica

Estas duas últimas características utilizarão uma abordagem diferente das demais, ao invés de ter uma família de valores já pré-definidos, estes serão adicionados pelo usuário do sistema, com isso pretende-se, neste trabalho, obter uma classificação mais adequada para a organização em questão. Os valores que estes atributos podem assumir serão vistos em detalhes quando falarmos sobre o cálculo de similaridade proposto por este trabalho (seção 4.4).

4.3 Análise da Caracterização

Após a selecionar os índices, é preciso analisar cada um deles e definir como cada um será representado no modelo proposto (seção 2.1). As seis características obtidas dos conceitos propostos no MAPS também têm sua representação baseada em valores propostos no mesmo.

- **Tamanho da equipe –** Esta é uma característica que tem impacto direto na forma de comunicação entre desenvolvedores do sistema [9], em pequenas equipes a comunicação informal é muitas vezes suficiente, porém quando se trata de equipes maiores é preciso utilizar padrões de comunicação bem definidos a fim de conseguir coordenar todos os seus integrantes de forma eficiente. Portanto, diversos riscos associados a problemas de comunicação estão diretamente relacionados ao tamanho da equipe de desenvolvimento.
    - **Valores –** Foi realizado um estudo e adaptação de valores de tamanho de equipes para a realidade brasileira, desta forma os valores sugeridos são:

I. **Muito pequena:** **1 – 6 Pessoas**
II. **Pequena:** **7 – 20 Pessoas**
III. **Média:** **21 – 50 Pessoas**
IV. **Grande:** **51 – 100 Pessoas**
V. **Muito grande:** **+100 Pessoas**

- **Distribuição Geográfica –** Em projetos com equipes geograficamente distribuídas é possível enfrentar dificuldades de transmissão de documentos além de outros problemas de comunicação entre seus membros, neste caso, a gerência de recursos em projetos desse tipo pode ser diretamente afetada.
    - **Valores –** Para esta característica temos os seguintes valores propostos:
        I. **Mesma sala;**
        II. **Mesmo prédio, diferentes salas;**
        III. **Mesma cidade, mesma empresa, prédios diferentes;**
        IV. **Mesma cidade, empresas diferentes;**
        V. **Cidades diferentes;**

- **Experiência da equipe de desenvolvimento –** Esta característica pode ser considerada diretamente um risco do projeto, e tem impacto direto no gerenciamento do mesmo, pois os prazos definidos sofrem variações de acordo com a experiência dos desenvolvedores. Por se tratar de uma característica complexa foi proposta uma decomposição em três características:
    - **Experiência no processo –** (número médio de projetos em que membros da equipe participaram onde foi utilizado o mesmo processo do projeto atual):
        I. **Nenhum projeto;**
        II. **1 projeto;**
        III. **2 a 3 projetos;**
        IV. **4 a 5 projetos;**
        V. **Mais de 5 projetos;**
    - **Experiência no domínio da aplicação –** (número médio de projetos em que membros da equipe participaram no domínio de aplicação do projeto atual):
        I. **Nenhum projeto;**
        II. **1 projeto;**
        III. **2 a 3 projetos;**
        IV. **4 a 5 projetos;**
        V. **Mais de 5 projetos;**

- **Experiência técnica –** (tempo médio de experiência da equipe com as principais tecnologias utilizadas no projeto atual):
  I. **Nenhum projeto;**
  II. **1 projeto;**
  III. **2 a 3 projetos;**
  IV. **4 a 5 projetos;**
  V. **Mais de 5 projetos;**

- **Tamanho do projeto –** Estudos mostram que estatisticamente projetos de grande porte têm menor probabilidade de sucesso. O tamanho do projeto está intimamente ligado à quantidade de atividades realizadas, portanto é possível que estas atividades tenham riscos associados.
  - **Valores –** Os valores sugeridos para projetos de software no Brasil são:
    I. **Até R$ 50.000,00**
    II. **Entre R$ 50.000,00 e R$ 150.000,00**
    III. **Entre R$ 150.000,00 e R$ 1.000.000,00**
    IV. **Entre R$ 1.000.000,00 e R$ 3.000.000,00**
    V. **Mais de R$ 3.000.000,00**

- **Tipo de Projeto –** Esta é uma característica vista como importante para a avaliação de riscos e não sugerida pelo MAPS, portanto adicionada à modelagem proposta por este trabalho a partir da seguinte premissa: projetos de software de um mesmo tipo são geralmente semelhantes. A partir desse conceito podemos inferir que este atributo é muito importante para a caracterização de projetos de software.
  - **Valores –** Atualmente uma grande quantidade de softwares surge diariamente para resolver diversos tipos de problemas e executar as mais diferentes tarefas. Portanto fazer uma taxonomia ou classificação de tipos de softwares não é uma tarefa simples, e mesmo que seja criada pode ser tornar inadequada para uma determinada organização ou ainda tornar-se obsoleta devido ao aparecimento de novos tipos. Sendo assim surge a questão: **Como criar uma classificação de softwares para uma**

organização específica? A modelagem proposta neste trabalho sugere que o usuário deve fornecer ao sistema quais tipos de software ele já desenvolveu em sua organização, caso eles existam. Não é difícil imaginar que uma empresa de desenvolvimento de softwares tenha um catálogo de produtos ou um *portfólio* com softwares de diferentes aplicações, portanto seus produtos já apresentam uma classificação naturalmente utilizada pela organização, sendo assim tem-se um bom referencial de tipos de projetos de software.

- **Tecnologia –** Este atributo define qual conjunto de ferramentas, linguagens e padrões de tecnologia foram utilizados no projeto, e.g. "Aplicação Web, com tecnologia Java[1] usando ambiente WSED[2]" ou "Aplicação desenvolvida em Delphi[3] usando banco de dados Oracle[4]". É fácil imaginar que a plataforma utilizada no projeto define diversas características do mesmo, portanto partimos da seguinte premissa: Projetos de software que utilizam uma mesma plataforma de desenvolvimento são em geral projetos semelhantes.
    - **Valores –** Devido a grande quantidade de ferramentas e plataformas de desenvolvimento seria difícil listar ou categorizar os seus valores. Além disso temos como um agravante o fato de que novas tecnologias e plataformas estão constantemente surgindo. Mesmo que ainda fosse possível listar todas as tecnologias e plataformas existentes, geralmente em uma mesma organização a quantidade de tecnologias utilizadas não é tão vasta. Portanto do mesmo modo que o atributo de tipo de software os valores referentes à plataforma tecnológica utilizados devem ser informados ao sistema pelo usuário na forma de conjunto de valores.

[1] Java, uma linguagem para desenvolvimento de softwares. http://java.sun.com.

[2] WSED (Websphere Studio Enterprise Developer), uma ferramenta para desenvolvimento de software: www.ibm.com.

[3] Delphi, uma ferramenta de desenvolvimento de software: www.borland.com.

[4] Oracle, um sistema gerenciador de banco de dados: www.oracle.com.

4.4 Cálculo de Similaridade e Recuperação de Casos

O cálculo de similaridade proposto neste trabalho é baseado na técnica de similaridade local (seção 2.2), onde as comparações serão feitas entre índices e não entre casos.

O modelo de cálculo de similaridade proposto neste trabalho se divide em duas partes principais, primeiro a similaridade feita atributo a atributo, em seguida é feita a aplicação dos pesos sobre os valores calculados anteriormente e então um valor de similaridade, em relação ao caso atual, será atribuído ao caso recuperado. A partir desse valor será escolhido o caso com maior similaridade em relação ao novo caso. A seguir, são apresentados os cálculos utilizados para os atributos utilizados no modelo proposto:

- **Atributos propostos no MAPS** – Os atributos: *tamanho da equipe*, *distribuição geográfica*, *experiência da equipe de desenvolvimento* e *tamanho do projeto* cujos valores já foram definidos (seção 4.3) terão sua representação feita através de símbolos ordenados (2.1.1) e a sua similaridade será calculada baseada na distância entre os mesmos através de uma medida numérica (2.2.1) que utiliza os valores de um (1) a cinco (5). A similaridade pode ser encontrada através de um cálculo simples, sendo 4 (quatro) similaridade máxima e 0 (zero) similaridade mínima.
- **Novos atributos** – Os atributos *tipo de* projeto e *plataforma tecnológica* serão representados através de um conjunto de símbolos não-ordenados, portanto vão poder assumir dois valores para o cálculo de similaridade: 0 (zero) ou 1 (um) representando similaridade mínima e máxima respectivamente.

O cálculo de similaridade entre dois atributos se dá através da seguinte fórmula:

$$S = (N - 1) - |Vp - Vc|$$

Equação 4.1 – Cálculo de Similaridade entre atributos.

Nesta equação ***S*** representa a similaridade entre os atributos, **N** representa a quantidade de valores que o atributo pode assumir, no caso dos atributos sugeridos pelo MAPS este valor corresponde a 5 (cinco). V_p e V_c representam os valores dos atributos pertencentes ao caso atual e ao caso da base de dados respectivamente.

Uma vez definida como será feita a comparação entre atributos, é usada uma fórmula que fornece um valor global de similaridade entre o caso atual e o caso recuperado da base de casos, conforme a Equação 4.2.

$$Sg = W_1 S_1 + W_2 S_2 + W_3 (S_{3.1} + S_{3.2} + S_{3.3})^2 + W_4 S_4 + W_5 S_5 + W_6 S_6$$

Equação 4.2 – Cálculo de Similaridade global.

Quanto maior o valor de S_g, maior a similaridade entre os casos, desta forma é possível criar um *ranking* de similaridade a partir dos casos da base de casos.

As variáveis S_1 à S_6 da Equação 4.2 representam as similaridades dos atributos obtidas anteriormente através da Equação 4.1. A variável S_3 representa o atributo *Experiência da Equipe de Desenvolvimento*, que por sua vez é dividida em três valores. A soma desses valores (experiência no processo, domínio e técnica) é elevada ao quadrado, desta forma obtemos uma maior distância entre uma equipe com nenhuma experiência (valor mínimo) e uma equipe que participou de mais de cinco projetos (valor máximo).

As variáveis W_1 à W_6 da Equação 4.2 representam pesos utilizados para aumentar a eficiência do modelo em organizações distintas, podendo ser usados para "calibrar" o cálculo de similaridade. Quando o valor de um peso é maior do que outro isto implica que a influência de um determinado atributo é maior do que a de outro no cálculo de similaridade global. Com isso as prioridades de uma organização podem ser adicionadas ao cálculo.

Portanto, se em uma organização, por exemplo, o mesmo grupo de pessoas é sempre utilizado para desenvolver aplicações de diversos domínios de conhecimento, então neste caso o peso do atributo *Tamanho da Equipe* e o do atributo *Experiência da Equipe de Desenvolvimento* podem ser menores do que o peso do atributo *Tipo de Projeto*.

Apesar do cálculo de similaridade proposto no modelo parecer simples, estudos mostram que em sistemas RBC cálculos complexos não necessariamente aumentam o desempenho do sistema. De fato o uso de técnicas mais simples em sistemas RBC é recomendado, pois apresentam um apelo intuitivo ao usuário final e à não-especialistas em geral [10].

A recuperação de casos em um sistema RBC tem como objetivo principal trazer, da base de casos, todos os casos que possam ser relevantes para o problema dado, portanto, a solução para o problema, caso exista na base de casos, deverá estar contida neste subconjunto formado pelos casos recuperados.

Neste trabalho a recuperação dos casos será feita utilizando a técnica de recuperação seqüencial (seção 2.3), em que fazemos uma comparação do problema com todos os casos da base de casos.

4.5 Protótipo

Um protótipo simples foi desenvolvido utilizando PHP e banco de dados MySQL com base no modelo de identificação de riscos apresentado neste trabalho. Através de poucos casos é possível demonstrar como o modelo pode ser utilizado na construção de um sistema de suporte à decisão para identificação de riscos de projetos de software.

Um modelo de dados relacional simples foi utilizado para armazenar as informações sobre os casos do sistema. Além dos casos, informações sobre contexto de um risco em determinados projetos também são armazenadas.

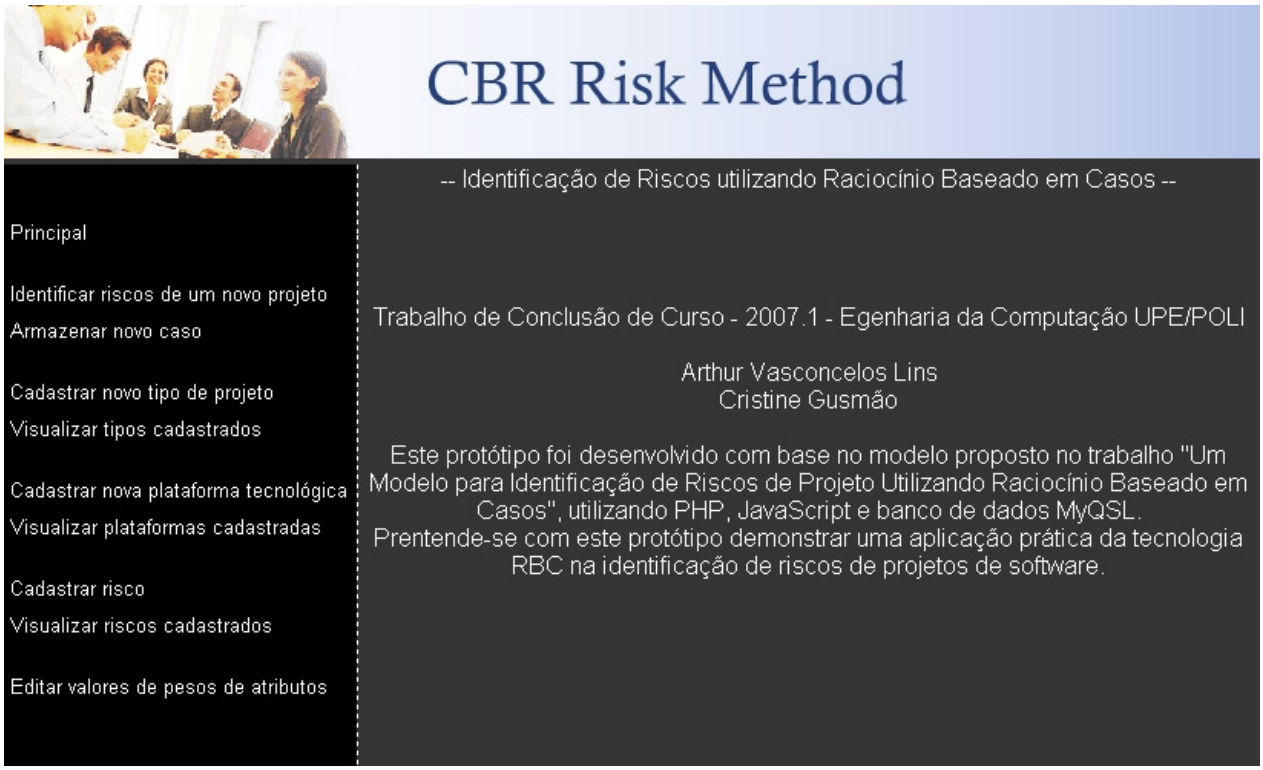

Figura 4.1 – Tela inicial do protótipo.

As principais funcionalidades desenvolvidas são:

- **Cadastrar e Visualizar tipos de projetos** – O atributo *Tipo de Projeto* tem seus valores definidos pelo usuário, como vimos na seção 4.3.
- **Cadastrar e Visualizar plataforma tecnológica** – O atributo *Plataforma Tecnológica* tem seus valores definidos pelo usuário, como vimos na seção 4.3.
- **Cadastrar e Visualizar Riscos** – Permite o cadastro de riscos identificados na base de dados.
- **Armazenar novo caso** – Permite a incorporação de um novo caso na base de dados, este caso de uso é divido em dois passos:

- O usuário fornece informações sobre um projeto de software, escolhendo valores para cada atributo, onde os valores para *Tipo de Projeto* e *Plataforma Tecnológica* precisam estar previamente cadastrados.
- Em seguida o usuário faz uma associação entre o caso que será armazenado e seus riscos, que foram previamente cadastrados no sistema. Informações sobre o contexto do risco no projeto e plano de contingência adotado também são inseridas pelo usuário.

- **Identificar riscos de um novo projeto** – Esta é a principal funcionalidade do protótipo, onde foi implementado o modelo matemático para calcular similaridade entre projetos de software. Neste caso de uso o é requisitado ao usuário que informe os valores de atributos do projeto, todos os casos da base de dados são comparados ao projeto informado, os três projetos considerados de maior similaridade tem seus riscos apresentados ao usuário, assim como as informações sobre contexto do risco no projeto e plano de contingência adotado.
- Editar valores de pesos de atributos – Através desta funcionalidade o usuário poderá definir quais valores serão utilizados como pesos no cálculo de similaridade. O uso dos pesos, mostrado na seção 4.4, permite que o usuário ajuste o cálculo de acordo com a sua organização, a fim de obter melhores resultados na identificação de riscos.

4.6 Resumo do Capítulo

Neste capítulo foi apresentado um modelo para identificação automática de riscos de projetos utilizando RBC. A proposta descrita inclui um modelo de representação de projetos de software, sua caracterização e cálculo de similaridade entre projetos, tanto em nível de atributos quanto em nível de casos.

O cálculo proposto para o modelo é flexível, podendo ser ajustado através de pesos de atributos utilizados no cálculo de similaridade, desta forma o sistema pode ser "calibrado" para atingir melhores resultados, através do ajuste dos pesos da Equação 4.2 dependendo das características organizacionais predominantes.

Ao final deste capítulo descrevemos um protótipo que utilizada um modelo relacional de dados para armazenar informações sobre projetos e riscos. A utilização do protótipo permitiu a avaliação dos requisitos para o modelo proposto.

Capítulo 5 – Conclusões e Trabalhos Futuros

Este trabalho tem como contribuição principal a proposta de uma técnica para identificação de riscos através do uso de raciocínio baseado em casos.

Um estudo realizado sobre modelos de Gerência de Riscos disponíveis na literatura, revelou que a identificação de riscos é comum à maioria dos modelos existentes, portanto acredita-se que este trabalho pode contribuir para a Gerência de Riscos de qualquer organização que utilize um desses modelos.

Adicionalmente, um estudo realizado sobre a técnica de RBC procurou identificar uma forma de apoio à atividade de identificação de riscos de projetos e conseqüentemente auxiliar na tomada de decisão gerencial através de uma identificação automática de riscos. Foi apresentado um modelo para representação e comparação entre projetos de software juntamente com um modelo de dados utilizado no protótipo do projeto.

5.1 Principais Contribuições

Um modelo inicial de representação de casos de projetos de software e cálculo de similaridade entre estes casos foi definido neste trabalho com base na literatura de RBC disponível. A partir deste modelo é possível construir um software com o objetivo de identificar riscos utilizando experiência passada.

É possível que, através de melhorias, este modelo possa ter maturidade e qualidade suficientes para fornecer uma base para o desenvolvimento de sistema com resultados significativos, podendo até mesmo ajudar gerentes de projetos inexperientes em uma determinada organização.

5.2 Trabalhos Futuros

Com base nos resultados analisados com a realização deste trabalho, além da constatação relacionada à literatura existente, algumas melhorias necessárias podem ser sugeridas. Assim, as seguintes direções surgem como perspectivas de trabalhos futuros.

- Avaliação – O modelo pode ser avaliado através de simulações utilizando um protótipo, estas podem ser feitas através das seguintes abordagens:
  - Executar o protótipo dentro do ambiente de uma determinada organização avaliando seus resultados.
  - Comparar os riscos identificados automaticamente pelo protótipo com os riscos identificados através de métodos encontrados na literatura de Gerência de Riscos, como exemplo o método apresentado na seção 3.3.
- Adição de informações – No cálculo de similaridade informações podem ser requisitadas ao usuário a fim de selecionar o melhor caso dentre um grupo de casos considerados similares.
- Ajustes no cálculo de similaridade – A partir dos resultados obtidos com testes e validações fazer as devidas melhorias no cálculo de similaridade utilizando os pesos da equação 4.2.

Além das melhorias identificadas, é possível sugerir também a integração com uma ferramenta já existente de Gestão de Riscos para Ambientes de Múltiplos Projetos chamada *m*PRIME[1].

O *m*PRIME foi desenvolvido como *add-in* para o Microsoft Project (MS Project), e pode ser utilizado em ambientes de múltiplos projetos. Sua definição teve por base estudos acadêmicos em nível de mestrado e doutorado do Centro de Informática da Universidade Federal de Pernambuco.

[1] *m*PRIME na Web: www.suppera.net

Referências Bibliográficas

[1] Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos (Guia PMBOK®) Terceira edição 2004 Project Management Institute.

[2] ABEL, M. *Raciocínio Baseado em Casos*. Instituto de Informática da Universidade Federal do Rio Grande do Sul – UFRGS p. 3. 1999.

[3] Expert Systems Development Series: Introduction to Case-based Reasoning. Diagnostic Strategies. 1999. Disponível em: www.diagnosticstrategies.com, último acesso em 20/02/2007.

[4] WANGENHEIM, C. G e WANGENHEIM, A. *Raciocínio Baseado em Casos*. Ed. Manole Ltda. São Paulo, Brasil. 2003.

[5] MOURA, H. P e GUSMÃO, C. e CORREIA, B.C.S. *Portfolio Management: A Critical View of Risk Factors Balancing*. Centro de Informática – Universidade Federal de Pernambuco (UFPE) CP 7851, Cidade Universitária, Recife, PE, Brasil.

[6] GUSMÃO, C.M.G. e MOURA, H. P. *Gerência de Risco em Processos de Qualidade de Software: uma Análise Comparativa*. In: III Simpósio Brasileiro de Qualidade de Software – Brasília, DF – 2004.

[7] CARR, M. et al. (1993) *Taxonomy Based Risk Identification*. Technical report CMU/SEI-93-TR-6. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. USA.

[8] GUSMÃO, C.M.G; MOURA, H. P. *Gestão de riscos para ambientes de múltiplos projetos de software: teoria e prática*. In: IV ERI MG – Escola Regional de Informática de Minas Gerais. ISBN 85-7669-048-9. Poços de Caldas – MG, 2005.

[9] COELHO, C.C. *MAPS: um Modelo de Adaptação de Processos de Software*. Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, Centro de Informática, Universidade Federal de Pernambuco, Recife. 2003

[10] Eman, E K e Benlarbi, S e Goel, N. *Comparing Case-based Reasoning Classifiers for Predicting High Risk Software Components*. Institute for Information Technology, National Research Council Canada. 2001.

[11] ABEL, M. *Um Estudo Sobre Raciocínio Baseado em Casos*. Universidade Federal do Rio Grande do Sul – UFRGS p. 9. 1996. Disponível em http://www.inf.ufrgs.br/gpesquisa/bdi/publicacoes/files/CBR-TI60.pdf. Último acesso em 20/06/2007.

Anexo A

Este anexo exibe a taxonomia proposta pelo SEI para identificação de riscos apresentada na Figura A.1, nem todos os elementos categorizados encontram-se na figura, para mais detalhes consultar o documento desenvolvido por Carr et al [7].

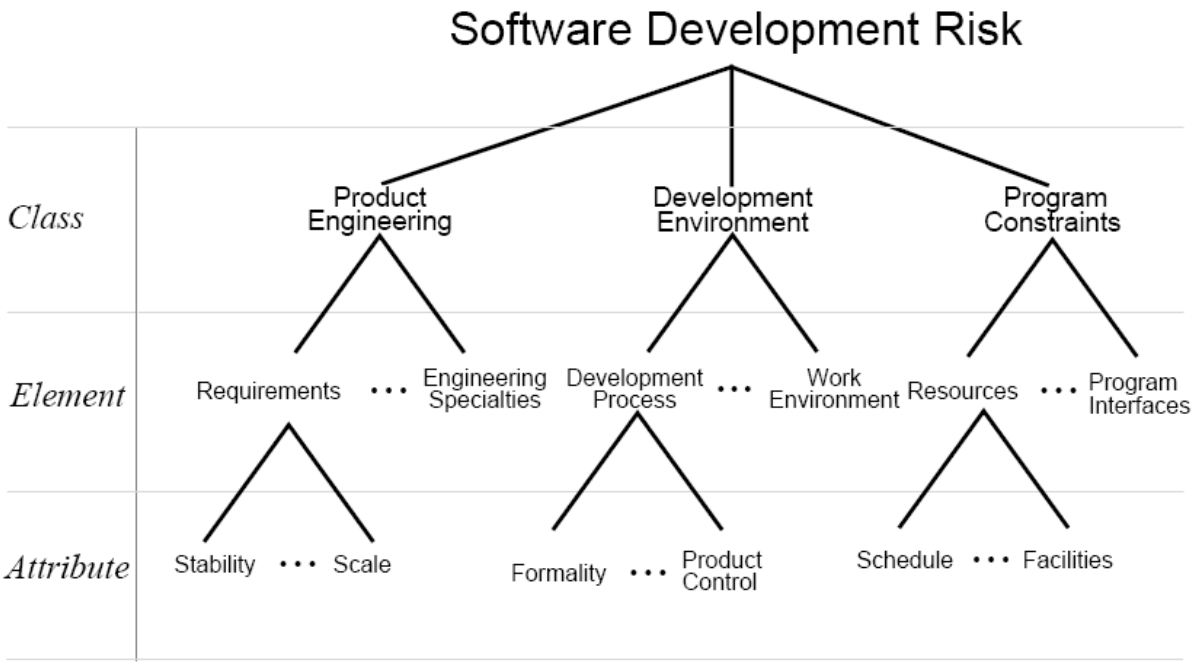

Figura A.5.1. Taxonomia de processos de desenvolvimento de software [7].