Um mecanismo de recuperação de design rationale em documentos na web

Lisandra Cazassa Fumagalli

Orientador: *Profa. Dra. Renata Pontin de Mattos Fortes*

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

"VERSÃO REVISADA APÓS A DEFESA"

Data da Defesa:	03/02/2005
Visto do Orientador:	

USP – São Carlos
Março/2005

"Conhecimento real
é saber a extensão
da própria ignorância."

(Confúcio)

Agradecimentos

Existem algumas pessoas a quem eu gostaria de agradecer.

Inicialmente aos meus pais, **José** e **Áurea**, pelo incentivo, compreensão e amor incondicional. À minha irmã, **Luciana** e ao meu "irmão" **Nenê** pelas conversas, pelo apoio e pelo carinho. Vocês são parte de mais essa conquista. Obrigada por tudo que vocês fizeram e fazem por mim. Eu os amo muito!!!

À **Renata**, por me orientar, acreditar em mim e, sobretudo, por me ensinar que o mais importante de um trabalho é o quanto podemos aprender com ele. Obrigada por sua amizade e pelo aprendizado.

Ao meu namorado, **Bruno**, por me ajudar a superar as crises, aumentar minha auto-estima e a confiança em mim mesma. Você é parte dessa minha vitória. Adoro muito você!

Às minhas grandes amigas **Eliz**, **Flávia Karla**, **Marina** e **Michele** pela amizade e por compartilharem comigo mais essa etapa da minha vida.

Ao pessoal do **Labes**, especialmente à **Debora** por me ajudar na revisão desta dissertação e por ser minha "vice-orientadora".

Ao **Rafael** (Pera) pela sua contribuição neste trabalho.

Ao **CNPq** e aos meus pais pelo apoio financeiro.

E principalmente a **Deus**, por me dar essa vida maravilhosa. Obrigada pela oportunidade de conviver com pessoas tão especiais e de realizar este trabalho.

Enfim, obrigada a todos por fazerem parte desse momento especial da minha vida e por dividirem comigo as tristezas e multiplicarem as alegrias!!!

Resumo

Design Rationale (DR) consiste em um conjunto de informações relacionadas ao processo de desenvolvimento e de tomada de decisão de um projeto. Nos projetos, em especial, adquirir e disponibilizar tais informações são práticas importantes para a melhoria das atividades de desenvolvimento e conseqüentemente da qualidade do produto desenvolvido. Por meio da atividade de documentação, os artefatos produzidos durante o projeto constituem a base para que ligações possam ser inseridas e expressem as relações com o DR correspondente. Um documento XML se apresenta como mecanismo apropriado para essa atividade pois além das informações, é possível acrescentar significado a essas informações. No entanto, a utilização efetiva das informações nesse documento XML só é possível se elas forem encontradas e exploradas de maneira a auxiliar os membros das equipes de projeto na realização de suas atividades. A ferramenta DocRationale foi desenvolvida para permitir o armazenamento e recuperação de informações de projeto de software, e respectivo DR. No entanto, para a exploração do DR armazenado, somente a navegação simples foi prevista inicialmente. Assim, a busca por informações de DR torna-se bastante custosa. Neste trabalho é apresentado um mecanismo para busca de DR, com o propósito de melhorar a exploração dessas informações.

Abstract

Design Rationale (DR) consists of a set of information related to the development and decision process of a project. In projects, acquiring and making such information available are important practices for the improvement of development process. Consequently the product has higher quality. With the documents produced during the software development is possible to create links with their corresponding DR. In that context, an XML document may be considered an appropriate mechanism for the documentation activity because information in those documents have a meaning. However, the effective use of information contained in this XML document is only possible if these information are searched and explored according to the needs of project teams. The DocRationale tool was developed to storage and recovery information related to software projects and their respective DR. However, as only the simple navigation was foreseen initially, searching for DR information is very onerous. In this work it is presented a mechanism for DR searching as a way to improve the exploration of these information.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO

1

Introdução

Na engenharia de software, o principal objetivo é produzir software com qualidade, reduzindo-se os custos e os esforços exigidos nas várias fases do desenvolvimento e na manutenção [Pressman, 2000]. Nesse sentido, técnicas e métodos se consolidam e contribuem para o reaproveitamento das experiências e a recuperação das decisões tomadas durante o processo de desenvolvimento de software. Além disso, organizações nessa área têm procurado utilizar o conhecimento adquirido como importante recurso para o cumprimento das atividades de engenharia de software [Borges & Falbo, 2002; Markkula, 1999; Wangenheim et al., 1999] e para obter melhorias no processo de desenvolvimento. Tais medidas contribuem para melhorar a qualidade dos produtos desenvolvidos. Assim, pode-se afirmar que a aquisição e busca de informações relacionadas às decisões de projeto se tornam necessárias para que as experiências sejam aproveitadas em projetos futuros.

Observa-se, entretanto, que mesmo quando os processos de desenvolvimento são conduzidos com qualidade, grande parte dos artefatos elaborados nas fases intermediárias é descartada, sendo somente registrada a solução escolhida para prosseguir o processo. Da mesma maneira, os argumentos que levaram à escolha de uma particular alternativa não são, em geral, explicitamente registrados ou documentados [Lamsweerde, 2000]. Alternativas consideradas como soluções, mas que foram posteriormente descartadas, também não são registradas. Em outras palavras, as informações relacionadas às decisões que ocorrem durante o processo de desen-

volvimento são perdidas pelo fato das mesmas não serem registradas. Nota-se, portanto, que não é possível retormar as decisões, assim como as alternativas envolvidas, fazendo com que os projetistas geralmente repitam experiências que poderiam ser evitadas.

Conclui-se, portanto, que as informações relacionadas às decisões de projeto são muito úteis para outros projetos, uma vez que erros podem ser evitados e alternativas anteriormente consideradas podem ser mais bem reaproveitadas [Souza et al., 1999], pois soluções discutidas e adotadas em um projeto podem ser relevantes a outros.

O *Design Rationale* (DR) se aplica nesse contexto, constituindo uma base de informações relacionadas às decisões tomadas e também às razões que propiciaram cada decisão, incluindo suas justificativas, alternativas consideradas, assim como o raciocínio empregado no desenvolvimento de um projeto [Lee, 1997; Gruber & Russell, 1991; Moran & Carroll, 1996; Regli et al., 2000].

A necessidade de armazenar as informações, tanto registradas nos documentos, quanto as relacionadas a DR, e possibilitar sua efetiva disponibilização para o compartilhamento do conhecimento relacionado à execução das atividades de engenharia de software motivou o desenvolvimento da DocRationale [Francisco et al., 2003; Francisco, 2004]. A DocRationale é uma ferramenta *Web* que tem por finalidade permitir o armazenamento e recuperação de informações de projeto, e respectivo DR.

No estágio atual, a DocRationale ainda possui deficiências. Neste trabalho, procura-se minimizar algumas delas e sugerir novas soluções que possam melhorá-las.

1.1 Objetivos

De maneira geral, o objetivo deste trabalho foi evoluir a ferramenta DocRationale. Para isso, o primeiro passo foi identificar suas deficiências e propor melhorias para torná-la mais interativa. Neste contexto, definiu-se o foco principal deste trabalho que foi investigar um mecanismo para recuperação do DR capturado e armazenado na DocRationale. A proposta deste mecanismo é ser mais objetivo e menos custoso que a abordagem de navegação utilizada na ferramenta atualmente para recuperar o DR.

1.2 Organização do Trabalho

Neste capítulo foram apresentados a contextualização, a motivação e os objetivos deste trabalho. No Capítulo 2 são apresentados os fundamentos sobre *Design Rationale*, abordando a definição, as perspectivas que podem ser adotadas nos sistemas de DR e os modelos de representação. A captura e a recuperação de DR também são mencionadas no segundo capítulo. No Capítulo 3 é mostrado o estado da arte sobre a recuperação de informação. São apresentadas abordagens de recuperação tanto para *Web* quanto para os sistemas de DR. A DocRationale é o tópico do Capítulo 4. São apresentadas a descrição, arquitetura e a maneira como as informações são apresentadas na ferramenta. Há também uma discussão sobre as deficiências atualmente apresentadas e sobre melhorias para minimizar esses problemas. As tecnologias consideradas para oferecer suporte à recuperação de DR na ferramenta DocRationale são apresentadas no Capítulo 5. O mecanismo investigado com o propósito de oferecer suporte a uma recuperação menos custosa na ferramenta DocRationale é apresentado no Capítulo 6. Finalmente, no Capítulo 7 são realizadas as conclusões deste trabalho de mestrado.

CAPÍTULO

2

Fundamentos sobre *Design Rationale*

2.1 Considerações Iniciais

Design Rationale (DR) se refere ao conjunto de informações relacionadas não só às decisões, mas também às razões que propiciaram cada decisão dentro de um projeto [Lee, 1997].

As pesquisas relacionadas a DR se concentram no aperfeiçoamento de métodos e representações apoiadas por computador para a captura, registro e reuso das decisões tomadas pelos projetistas durante o processo de projeto [Peña-Mora et al., 1995; Karsenty, 1996]. Um dos grandes desafios é tornar o esforço de registrar o DR menos oneroso aos projetistas, mas de maneira que esse registro seja suficientemente estruturado para permitir futura recuperação.

A utilização de DR é bastante apropriada quando o objetivo é reaproveitar experiências, retomar o processo de decisão e reutilizar soluções.

A definição, as perspectivas e os modelos de representação de DR são abordados neste capítulo. A captura, o armazenamento e a recuperação de DR também são brevemente descritos.

2.2 Definição

O termo *Design Rationale* possui significados diferentes para diferentes pessoas. Para algumas representa uma argumentação, para outras implica em documentação de projeto, também é descrito como a captura e o potencial reuso da comunicação do projeto [Shipman & McCall, 1997].

Na literatura, diversas definições são dadas para DR. Segundo Gruber & Russell [1991], DR é a explicação de "como" e "porque" um artefato ou alguma parte dele é projetado de determinada maneira. Para Burge & Brown [2000], DR é o conjunto das decisões tomadas no decorrer do processo de projeto e das razões que propiciaram essas decisões. De acordo com Regli et al. [2000], DR é a justificativa relacionada à tomada de decisão para o projeto de um artefato ou parte dele, incluindo as deliberações, razões, alterações e todas as decisões intermediárias.

Considerando as definições mencionadas, pode-se concluir que DR consiste nas informações que permitem descrever o raciocínio dos projetistas para justificar as decisões do projeto resultante [Gruber & Russell, 1991]. Em outras palavras, representa o conhecimento de projeto, ou seja, é constituído de informações que se apóiam nas diferentes alternativas consideradas no processo de tomada de decisão para o desenvolvimento de cada artefato de um projeto.

Considerando suas propostas e características, grande número de benefícios podem ser obtidos com a utilização de DR. Pode-se citar:

- Proporcionar maior visibilidade das idéias, propiciando uma metodologia estruturada às decisões e assegurando que as decisões não estejam sendo tomadas de mancira arbitrária [Monk et al., 1995].
- Ser utilizado para verificar se o projeto atende os requisitos, e também para avaliar possíveis inconsistências no projeto [Burge & Brown, 2001].
- Ser utilizado para avaliar possíveis conseqüências de alterações no sistema e ajudar a decidir se o mesmo deve ser adaptado ou alguma parte dele deve ser refeita [Monk et al., 1995].
- Auxiliar a compreensão de "como" e "porque" o sistema foi projetado de tal maneira; assim, durante a fase de manutenção, por exemplo, esse entendimento pode minimizar

possíveis esforços e potenciais erros na realização das modificações [Burge & Brown, 2001].

- Ser utilizado para determinar que partes do projeto podem ser reutilizadas e, em alguns casos, sugerir "onde" e "como" o projeto deveria ser modificado a fim de atender novos requisitos [Monk et al., 1995].

- Ser utilizado para documentar o projeto, oferecendo um resumo do histórico do projeto e as razões das decisões e escolhas. Também pode fornecer uma visão geral do produto final, permitindo a descoberta de erros no desenvolvimento do projeto [Burge & Brown, 2001; Souza et al., 1999].

- Auxiliar a comunicação entre os membros de uma equipe de projeto, entre projetistas e usuários, e entre equipes atuais de projeto e outras que possam construir projetos similares ou reusar partes de projetos já desenvolvidos [Monk et al., 1995].

Apesar de todas as vantagens mencionadas, raramente os sistemas de DR têm sido adotados na prática. Observa-se que os problemas relacionados ao uso de DR não estão centrados especificamente na captura, representação ou recuperação. Os obstáculos a serem transpostos dependem, de certa maneira, da perspectiva sobre DR adotada no sistema.

Existem três perspectivas distintas sobre DR: argumentação, comunicação e documentação. Cada uma dessas perspectivas utiliza estratégias que influenciam a habilidade de cada uma delas capturar, armazenar e recuperar os DRs [Shipman & McCall, 1997]. Possibilitar mais flexibilidade na captura de DR durante a documentação é um exemplo de obstáculo a ser superado. As perspectivas e os problemas de cada uma delas são apresentados a seguir.

2.3 Perspectivas sobre Design Rationale

Segundo Shipman & McCall [1997] existem, no mínimo, três perspectivas sobre DR: argumentação, comunicação e documentação. Os objetivos que as perspectivas procuram atingir são diferentes para cada uma delas, e para alcançá-los, abordagens diversas podem ser utilizadas. No entanto, ainda que sejam diferentes, tais perspectivas não são contraditórias e podem ser utilizadas combinadas, definindo as abordagens híbridas.

Perspectiva de Argumentação

O propósito de registrar DR, a partir da perspectiva de argumentação, é destacar as deficiências de um projeto e, então, corrigi-las. Essa perspectiva é geralmente adotada por projetistas que procuram melhorar a qualidade dos projetos aperfeiçoando os argumentos utilizados pelos próprios projetistas [Shipman & McCall, 1997].

Na perspectiva de argumentação, o DR a ser capturado deve ser adequadamente representado em modelos de representação. Um problema observado nesse tipo de perspectiva é a grande resistência dos projetistas ao serem obrigados a utilizar tais esquemas enquanto realizam suas atividades de projeto. Nesse sentido, a captura da argumentação é facilitada se um especialista transcrever os argumentos dos outros membros da equipe no esquema de representação [Conklin & Yakemovic, 1991].

Embora a captura de DR seja considerada problemática quando a perspectiva de argumentação é utilizada, a recuperação do DR é bastante facilitada. O fato do DR ser capturado de maneira estruturada, possibilita sua recuperação efetiva [Shipman & McCall, 1997].

Perspectiva de Comunicação

O objetivo da perspectiva de comunicação é registrar toda a comunicação natural da equipe sem que haja interferência no processo de projeto [Shipman & McCall, 1997]. O registro dessa comunicação pode incluir gravações de reuniões e conversas, e-mails, desenhos e registros feitos em papel ou em quadros.

Os problemas observados nesse tipo de perspectiva são a falta de estruturação na captura do DR e o registro de informações não necessariamente pertencentes ao processo de projeto. Como a discussão entre os projetistas é, em geral, realizada de maneira livre, desordenada e com muitas divagações, torna-se difícil estruturar o DR capturado e selecionar as informações importantes [Shipman & McCall, 1997].

A imposição mínima ou inexistente de restrições, adicionada a não exigência de estruturação e representação, evidencia-se numa captura trivial. Por outro lado, a recuperação efetiva do DR registrado é bastante complexa. Como dificuldade inicial, toda a comunicação capturada deve ser convertida para um meio digital. Essa conversão é necessária mas insuficiente para a recuperação efetiva do DR [Shipman & McCall, 1997]. Assim, apesar de não apresentar os problemas com a captura, como na perspectiva de argumentação, a perspectiva de comunicação

apresenta desvantagens quanto à recuperação.

Perspectiva de Documentação

Na perspectiva de documentação, o propósito é documentar os resultados da argumentação, acrescida da explicação e das razões para esses resultados. Informações adicionais como alternativas consideradas e discussões não são informações relevantes [Shipman & McCall, 1997]. Dessa maneira, é uma característica da perspectiva de documentação registrar menos informações quando comparada às perspectivas de argumentação e comunicação.

Dentre as perspectivas sobre DR, a de documentação é a mais aceita entre os projetistas. A principal razão para isso é que, de um jeito ou outro, os projetistas são obrigados a registrar suas funções e ações no processo de projeto. Outra razão para a aceitação, deve-se ao fato da criação dos documentos ser realizada após o processo de decisão [Shipman & McCall, 1997].

O tempo e o esforço gastos na geração dos documentos é uma das principais desvantagens da perspectiva de DR. Recursos que poderiam ser utilizados em outras atividades do projeto, devem ser dispensados na documentação. Outra desvantagem é a documentação imprecisa das decisões. Particularmente, a documentação possui pouco valor se as decisões não forem claramente descritas, explicadas e justificadas.

Devido aos fatores mencionados anteriormente, a recuperação do DR armazenado é restrita. Como parte das informações é perdida ao serem registradas, a recuperação do DR também é prejudicada.

Na Tabela 2.1 é apresentado um resumo sobre as características da captura e da recuperação nas perspectivas sobre DR.

Tabela 2.1: Características da captura e recuperação nas perspectivas sobre DR

Perspectiva	**Captura**	**Recuperação**
Argumentação	estruturada	facilitada
Comunicação	ampla e livre	dificultada
Documentação	sintetizada	restrita

Com o objetivo de superar as limitações de cada uma das perspectivas sobre DR mencionadas, sistemas que as combinam têm sido propostos. A finalidade dos sistemas de abordagem híbrida é aproveitar as vantagens de cada uma das perspectivas, complementando-as. Em geral, isso significa disponibilizar a captura formal de informação, facilitando seu armazenamento e

recuperação, mas de maneira que as atividades de captura e recuperação sejam mais flexíveis.

Ainda que a perspectiva sobre DR adotada em um sistema afete diretamente a facilidade com que serão realizadas a captura e a recuperação de DR nesse sistema, existem modelos de representação que auxiliam essas tarefas em relação ao DR [Regli et al., 2000].

2.4 Modelos de Representação de DR

Um bom modelo de representação é vital para permitir que as informações relacionadas ao DR dos projetos possa ser capturada, estruturada, recuperada e reutilizada de maneira efetiva.

As representações de DR podem ser formais, como regras de representação embutidas em um sistema de DR, ou mesmo informais, como gravações de áudio e vídeo. No entanto, registrar as informações utilizando representações semi-formais é o ideal, pois além de oferecer algum automatismo, ainda é compreensível aos usuários que provêem as informações [Burge & Brown, 2000].

Representações semi-formais são geralmente utilizadas para representar a argumentação [Burge & Brown, 2000]. As diferenças entre os modelos recai sobre as entidades escolhidas para capturar os argumentos que, por sua vez, determinam os tipos e o nível de detalhes dos argumentos a serem capturados [Shum & Hammond, 1994]. De qualquer maneira, os objetivos dos modelos de representação de DR nos projetos são [Shum & Hammond, 1994; Isenmann & Reuter, 1997]: (i) Estruturar problemas de projeto; (ii) Manter a consistência no processo de tomada de decisão; (iii) Manter o resumo das decisões; (iv) Comunicar as justificativas de projeto; (v) Registrar cronologicamente o processo de projeto; (vi) Oferecer suporte à construção cumulativa de conhecimento de projeto, através do reuso de DR.

Considerando os modelos de representação para a argumentação, os quatro mais comuns são: IBIS (*Issue-Based Information System*), PHI (*Procedural Hierarchy of Issues*), QOC (*Questions, Options and Criteria*) e DRL (*Design Rationale Language*).

2.4.1 IBIS - Issue-Based Information System

O modelo de representação IBIS se baseia no princípio que o processo de projeto para problemas complexos é fundamentalmente uma "conversa" entre os projetistas. Através dessa conversa os projetistas podem expor suas experiências e pontos de vista relacionados às decisões

do projeto [Conklin & Begeman, 1988].

No modelo IBIS a representação é realizada através de três entidades: **Questões** (*Issues*[1]), **Posições** (*Positions*) e **Argumentos** (*Arguments*). As Questões representam qualquer problema ou pergunta e podem possuir Posições e outras Questões associadas a elas. As Posições são declarações ou afirmações que resolvem uma Questão. Cada Posição, por sua vez, também pode se relacionar com um ou mais Argumentos, que podem ser contra ou a favor da Posição. Resumidamente, uma Questão é a raiz de uma árvore cujos os filhos são as Posições que possuem como filhos os Argumentos [Conklin & Begeman, 1988].

Em outras palavras, as entidades são declarações e outras contribuições que são geradas ou consideradas durante uma discussão. Tal discussão tem o propósito de debater as questões levantadas, assumindo que sua discussão e solução são relevantes para a solução de um problema do projeto [Isenmann & Reuter, 1997].

Além das entidades, existem os *links* que relacionam as Questões, Posições e Argumentos, por exemplo os *links* "especializa", "questiona", "substitui", "responde a", "apóia" etc. As entidades e os *links* do modelo de representação IBIS são mostrados na Figura 2.1.

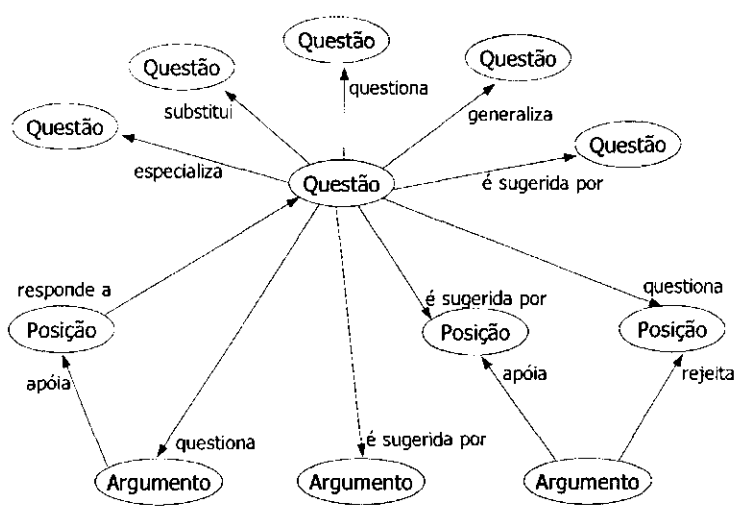

Figura 2.1: Modelo de representação IBIS (Adaptado de Conklin & Begeman [1988])

Segundo Conklin e Begeman [1988] a utilização do modelo de representação IBIS torna as reuniões mais produtivas pois as discussões não são desviadas dos assuntos originais. Outro ponto positivo do modelo é possibilitar que as equipes examinem o conjunto de questões

[1]Neste trabalho, para a palavra *Issue* foi adotada a tradução "Questão". No entanto, seu significado não se restringe necessariamente a uma pergunta. O sentido adotado é de um assunto a ser discutido.

não resolvidas e decidam qual a melhor ordem para discuti-las. No entanto, para usufruir dessas vantagens é necessário que os projetistas aceitem a argumentação e o modelo IBIS como instrumentos de suporte ao processo de tomada de decisão dentro de um projeto [Wiegeraad, 1999].

Algumas limitações também se destacam com a utilização do modelo. O IBIS não possui uma entidade separada para os requisitos e restrições impostas a uma solução, ou uma entidade que expresse qual é o objetivo da solução. Todos esses critérios são capturados implicitamente na entidade **Argumento** [Wiegeraad, 1999].

O IBIS possui um extenso conjunto de *links*, mas apesar dessa característica oferecer um meio fácil para unir as informações às discussões, a representação do IBIS não comporta muitas Questões, Posições e Argumentos.

Outro problema em relação à utilização do IBIS é a dificuldade, por parte dos projetistas, de exporem seus pensamentos em unidades discretas (Questão, Posição, Argumento), especialmente quando o problema não foi bem entendido e as idéias são vagas e confusas [Conklin & Begeman, 1988].

Outros modelos de representação similares ao IBIS foram desenvolvidos. Cada um deles tem como objetivo solucionar as restrições dos outros modelos.

2.4.2 PHI - Procedural Hierarchy of Issues

O modelo de representação PHI foi desenvolvido com o propósito de resolver as limitações do modelo IBIS. Algumas modificações foram realizadas na estrutura do modelo, sendo a entidade Posição do modelo IBIS alterada para Resposta e as relações entre as entidades simplificadas, utilizando-se apenas o *link* "serve". Assim, as entidades do modelo PHI são **Questões** (*Issues*), **Respostas** (*Answers*) e **Argumentos** (*Arguments*).

Outra modificação no modelo foi a criação de novos tipos de entidades: **Sub-Questão**, **Sub-Resposta** e **Sub-Argumento**. Cada Sub-Questão acrescenta novos questionamentos em relação à Questão. Cada Sub-Resposta representa uma parte mais específica da solução se comparada à sua respectiva Resposta. Cada Sub-Argumento adiciona detalhes ao seu Argumento relacionado. Assim, pode-se dizer que essas sub-relações expressam níveis de especificidade e granularidade em relação às suas respectivas entidades [Wiegeraad, 1999]. Além de utilizar a argumentação para lidar com as questões, o modelo PHI também oferece a possibilidade da

decomposição, por exemplo, uma Questão pode ser dividida em diversas Sub-Questões [Regli et al., 2000; Wiegeraad, 1999].

Observando a Figura 2.2, é fácil reconhecer a estrutura quase hierárquica do modelo PHI. Uma das vantagens da estrutura quase hierárquica do PHI é permitir ao usuário ver quais Questões são mais importantes e quais são mais detalhadas. Entretanto, essa estrutura não mostra como ela foi "crescendo" com o tempo. Outra vantagem é poder ter sua estrutura apresentada de em um formato linear ainda que entidades que possuam relação com mais de uma entidade precisem ser repetidas [Wiegeraad, 1999; Regli et al., 2000].

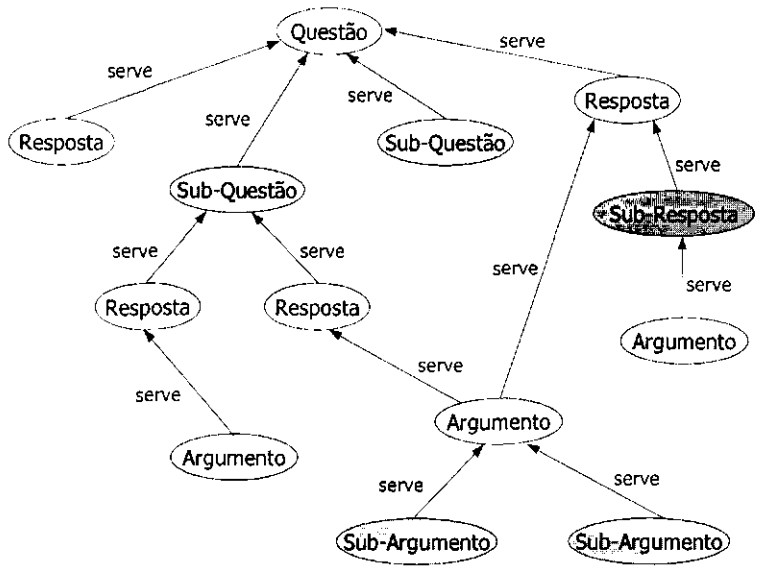

Figura 2.2: Modelo de representação PHI

As extensões no PHI melhoram a expressividade da representação do projeto quando utilizadas como um modelo para estruturar o processo de tomada de decisão. Entretanto, ainda há limitações na sua representação, uma vez que relacionamentos importantes ainda não podem ser facilmente expressados, como os relacionamentos entre questões [Lu et al., 1999].

2.4.3 QOC - Questions, Options and Criteria

O modelo QOC representa o processo de tomada de decisão dos projetos como uma rede de **Questões** (*Questions*) que enfatiza os problemas de projeto, **Opções** (*Options*) que representam possíveis soluções às Questões e **Critérios** (*Criteria*) que representam as razões contra ou a favor em relação às Opções [Burge & Brown, 2000; McKerlie & MacLean, 1993a].

Cada Questão é ligada a várias Opções através do *link* "responde a", como pode ser visto na Figura 2.3. As Opções são avaliadas positiva ou negativamente pelos Critérios. Além disso, uma Opção pode gerar outras Questões, assumindo que a Opção é parte do contexto de projeto de discussão futura [McKerlie & MacLean, 1993b].

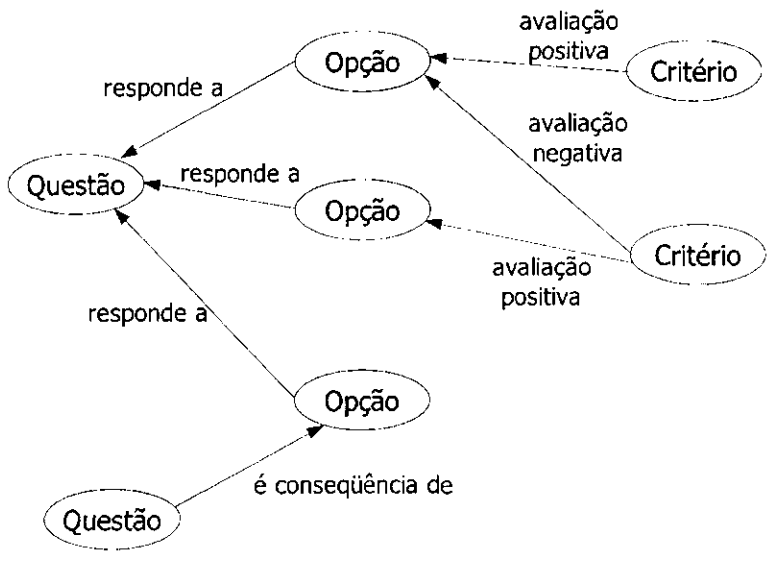

Figura 2.3: Modelo de representação QOC

Diferentemente dos modelos IBIS e PHI que privilegiam o tempo, o QOC enfatiza o espaço de projeto. Em outras palavras, o modelo QOC incentiva os projetistas a focalizarem um espaço de possíveis soluções ao invés de se limitarem à exploração de uma única solução. Assim, os projetistas têm que organizar, avaliar e criticar as informações de projeto e desenvolver soluções aceitáveis [Lu et al., 1999; McKerlie & MacLean, 1993a]. Há como resultado, um acúmulo de idéias e um entedimento do problema de modo sistemático [McKerlie & MacLean, 1993a; Wiegeraad, 1999].

Experimento realizado por McKerlie e MacLean [1993] evidencia que o uso do QOC ajuda a melhorar o entendimento das decisões e assiste a comunicação dentro da equipe de projeto. No entanto, o esforço gasto para produzir as Questões, Opções e Critérios é um dos problemas apontados. Outro problema mencionado é a falta de *guidelines* para determinar quais questões registrar e quando parar de refinar o espaço de projeto.

2.4.4 DRL - Design Rationale Language

DRL é uma extensão do modelo IBIS, cujo objetivo é melhorar a expressividade da representação através de um maior número de entidades e de relacionamentos entre elas [Lu et al., 1999].

Na figura 2.4 podem ser vistas todas as entidades desse modelo de representação e seus relacionamentos. As **Alternativas** (*Alternatives*) representam as opções dentre as quais se pode escolher. Os **Objetivos** (*Goals*) representam as propriedades que uma solução ideal deve ter. O **Problema de Decisão** (*Decision Problem*) representa o problema de escolher a Alternativa que satisfaça os Objetivos. As **Alegações** (*Claims*) representam os argumentos contra ou a favor às Alternativas. As **Questões** (*Questions*) podem ser propostas pelas Alegações. Os **Procedimentos** (*Procedures*) representam possíveis respostas às Questões ou podem ser resultados a partir de uma Alegação. Existem ainda os **Grupos** (*Groups*) que representam um conjunto de possíveis respostas a uma Questão [Lee, 1990].

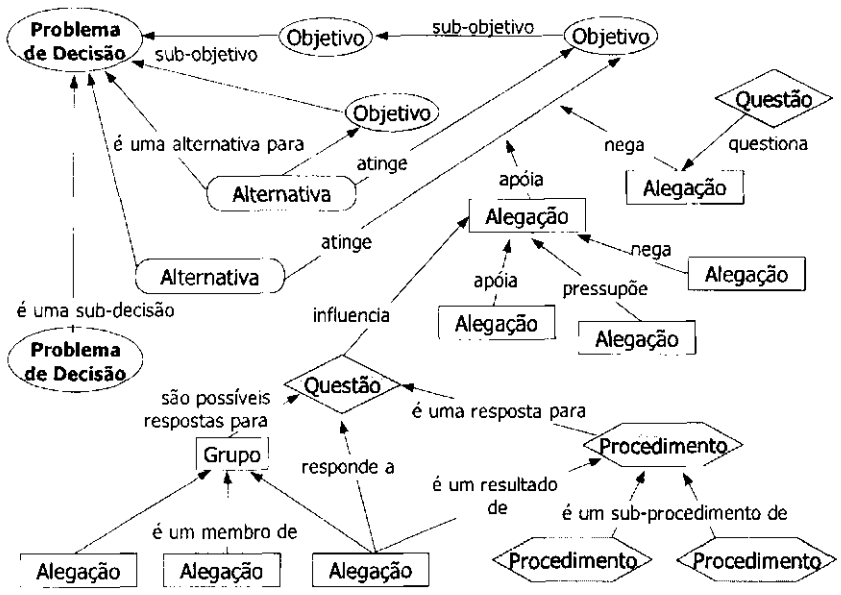

Figura 2.4: Modelo de representação DRL

A argumentação do DRL inicia com o Problema de Decisão. Para esse problema podem ser definidos diversos Objetivos que procuram resolvê-lo. Alternativas para tais Objetivos também são determinadas [Monk et al., 1995]. Para essas Alternativas surgem Alegações (contra ou a favor) para as quais, por sua vez, podem surgir Questões ou outras Alegações. Um Grupo de Alegações pode responder às Questões, assim como Alegações também o podem fazer.

Um propósito adicional do DRL é tentar gerenciar todas as restrições, objetivos e requisitos que são utilizados para a tomada de decisão separadamente da resolução das questões [Wiegeraad, 1999].

Para todos esses modelos de representação, diferentes termos são utilizados. No entanto, esses termos representam conceitos similares. Na Tabela 2.2 são mostrados os modelos e seus respectivos termos para conceitos como problema, solução, argumento e critério.

Tabela 2.2: Diferentes termos para conceitos similares nos modelos de representação de DR

Modelo	*Problema de Projeto*	*Solução*	*Argumento*	*Critério*
IBIS	Questão (*Issue*)	Posição	Argumento	
PHI	Questão(*Issue*)	Resposta	Argumento	
QOC	Questão (*Question*)	Opção	Critério	
DRL	Problema de Decisão	Alternativa	Alegação	Objetivo

Além das escolhas da perspectiva e da abordagem para a estruturação do DR, no desenvolvimento dos sistemas DR também é preciso considerar se eles serão sistemas orientados a processo ou orientados a característica.

2.5 Sistemas Orientados a Processo ou a Características

Sistemas orientados a processo enfatizam o DR como uma história do processo de projeto. O DR é meramente descritivo [Regli et al., 2000].

A abordagem orientada a processo tem sido mais empregada em domínios nos quais os problemas são vagos, a solução não é bem entendida ou quando há pouca, ou não há, padronização dos artefatos projetados [Conklin & Yakemovic, 1991]. Essa abordagem pode auxiliar os projetistas fornecendo informações históricas. No entanto, traduzir essas informações em representações que possam ser entendidas e processadas por computadores não é uma tarefa trivial [Regli et al., 2000].

Nos sistemas orientados a característica, a geração de DR é geralmente apoiada por bases de conhecimentos e as representações de DR são freqüentemente formais. Entretanto, partes de DR como "quem", "quando" e "porque" não podem ser tratadas por essa abordadem [Regli et al., 2000].

Quando a abordagem orientada a característica é adotada, os sistemas de DR são geralmente incluídos em sistemas de projeto. Isso auxilia os projetistas a registrarem o DR em um formato

formal que pode ser processado automaticamente. Dessa maneira, os sistemas podem apoiar os processos de projeto, por exemplo, a avaliação das decisões de projeto e a resolução de conflitos [Regli et al., 2000].

A combinação de ambas as abordagens pode ser adotada nos sistemas de DR a fim de contornar as limitações de cada uma.

2.6 Captura e Recuperação de DR

No desenvolvimento de sistemas de DR, a captura é um ponto crítico. Determinar quais informações capturar durante o projeto e como capturá-las é um problema fundamental dos sistemas de DR. Além disso, deve-se capturar a quantidade adequada de informações, pois essa medida influencia diretamente no sucesso dos sistemas [Regli et al., 2000].

O processo de captura de DR geralmente consiste em dois passos: obtenção e construção de DR. O primeiro envolve capturar tanta informação quanto possível durante o processo de projeto. O segundo trata da extração, organização e armazenamento da informação [Regli et al., 2000].

A captura de DR deve ser realizada durante todo o processo de projeto, registrando-se os argumentos, as alternativas consideradas e as decisões tomadas, que serão posteriormente estruturados e armazenados em um repositório [Lee, 1997].

Segundo Hu et al. [2000], pode-se dividir os métodos de captura de DR em duas categorias: a que requer intervenção do usuário e a que é executada automaticamente pelo sistema.

Nos sistemas que utilizam a abordagem com intervenção do usuário, os projetistas têm que registrar as informações de projeto manualmente. A documentação é o método mais utilizado quando essa abordagem é adotada no sistema [Regli et al., 2000]. O principal objetivo da documentação é registrar o histórico das atividades de processo, isto é, registrar as decisões tomadas [Shipman & McCall, 1997]. A desvantagem da documentação é que apenas as decisões são registradas, sem as justificativas que levaram a essas decisões.

No entanto, existem sistemas que utilizam mecanismos que possibilitam aos usuários proverem suas próprias expressões sobre as decisões de projeto e suas razões [Regli et al., 2000].

Nos sistemas que utilizam a abordagem automática, assume-se que existe um método para capturar a comunicação entre os projetistas da equipe de projeto. Essa comunicação registrada

pode ser utilizada para extrair o DR e as decisões envolvidas no processo de projeto [Shipman & McCall, 1997].

A captura automática pode incluir gravação de vídeo e de conversas telefônicas, e-mail, aplicações compartilhadas, assim como os desenhos e documentos trocados entre os projetistas. Nenhum esforço adicional é exigido dos desenvolvedores, que continuam a realizar suas atividades usuais. Quando a comunicação é arquivada digitalmente, as atividade de projeto podem ser processadas e o DR pode ser determinado [Regli et al., 2000]. A desvantagem dos sistemas que utilizam a abordagem orientada a característica é que a informação registrada não é estruturada, podendo dificultar o processo de recuperação.

A recuperação de DR é determinada pelo modelo de representação adotado para a captura e pelo domínio em que a informação é utilizada [Regli et al., 2000]. Diferentes abordagens podem ser utilizadas na recuperação de DR. Dentre elas se destacam: navegação, uso de *triggers*, recuperação baseada em consulta e estratégias de recuperação híbridas.

2.7 Considerações Finais

Design Rationale é a descrição das razões que justificam o projeto resultante [Gruber & Russell, 1991]. O uso de DR oferece diversas vantagens, por exemplo, auxiliar a compreensão, a manutenção de um projeto e a comunicação da equipe. Para isso, no entanto, é necessário definir que perspectiva e representação de DR adotar para possibilitar a sua efetiva captura e recuperação.

Neste capítulo foram apresentados alguns fundamentos de DR, incluindo definição, perspectivas e modelos de representação. Também foi apresentado sucintamente sobre a captura, o armazenamento e a recuperação de DR.

Como a recuperação de DR representa grande parte da investigação para o mecanismo de recuperação proposto neste trabalho, a revisão da literatura sobre a recuperação, tanto na *Web* como nos sistemas de DR, é apresentada no próximo capítulo.

CAPÍTULO

3

Recuperação de *Design Rationale*

3.1 Considerações Iniciais

A recuperação de informação consiste em um processo que envolve a representação, o armazenamento, a busca e a descoberta de informação considerada relevante por um usuário. Tradicionalmente, a informação se encontra na forma textual, de maneira que recuperar informações significa recuperar textos e documentos [Ingwersen, 1992]. O advento dos hiperdocumentos e, principalmente, o crescimento da *World Wide Web* (ou simplesmente *Web*) alavancaram o desenvolvimento de diversas tecnologias para a recuperação de informações.

A recuperação de informações no contexto de DR é, indiscutivelmente, um dos pontos essenciais para seu uso efetivo. A recuperação nos sistemas de DR tem como objetivo possibilitar que informações relevantes, anteriormente armazenadas, sejam buscadas e reutilizadas de modo que novos problemas sejam solucionados e até mesmo evitados, e que soluções sejam reaproveitadas [Regli et al., 2000]. Diferentes tecnologias de recuperação podem ser utilizadas nos sistemas de DR. Algumas dessas tecnologias serão apresentadas neste capítulo.

3.2 Tecnologias de Recuperação de Informação

Pode-se dizer que a *Web* é uma coleção de milhares de hiperdocumentos heterogêneos e distribuídos, conectados por *links*. A estrutura de um hiperdocumento é composta basicamente por uma rede de nós e *links*. Os nós são as porções de informação e o *link* faz a ligação entre os nós. A característica principal do hiperdocumento é o fato dele não possuir uma estrutura linear como os documentos convencionais, ou seja, o hiperdocumento apresenta ao usuário diversas opções, possibilitando que o mesmo determine a seqüência de leitura do texto [Nielsen, 1995].

Existem diversas tecnologias para a recuperação de informações na *Web*. Dentre elas, a navegação é a mais comum. Alguns tipos de navegação, os mais comumente citados na literatura, são apresentados a seguir.

3.2.1 Navegação Tradicional

A navegação é mais recomendada quando os espaços de informação são pequenos e familiares aos usuários, pois pequenos espaços de informação podem ser percorridos exaustivamente e espaços conhecidos são navegados sem dificuldades [Nielsen, 1995]. Dentre os diferentes tipos de navegação destacam-se:

- Navegação por *links*: é a maneira mais tradicional de navegação, mostrada na Figura 3.1e. Pode ser realizada entre *links* de duas ou mais páginas *Web*, ou através de *links* entre partes de uma mesma página [Nielsen, 1995].

- Navegação por *tour* guiado: nesse tipo de navegação, o usuário busca as informações desejadas através de um caminho de nós sugerido. Geralmente, o usuário pode modificar esse caminho, acrescentando nós ou alterando sua ordem. O *tour* guiado também pode ser suspenso e reiniciado no ponto em que foi interrompido. No entanto, ao mesmo tempo em que o *tour* guiado pode facilitar a navegação, a apresentação seqüencial da informação é retomada, descaracterizando a proposta de exploração aberta do espaço de informação [Nielsen, 1995].

- Navegação entre páginas precedentes e subseqüentes: a partir de uma página é possível optar por retornar à página anterior ou prosseguir à página seguinte [Nielsen, 1995]. Pode-se ter, como na Figura 3.1a, botões que oferecem esse tipo de navegação.

- Navegação através de facilidades de *design*: algumas facilidades de navegação são oferecidas pela própria página, por exemplo, índices, logotipos, barras de navegação, *frames*, cores dos *links* [Xu et al., 2001]. Na Figura 3.1d é mostrado um exemplo de índice presente na página *Web*.

- Navegação através de ferramentas disponíveis nos *browsers* padrão: os *browsers* oferecem diversas ferramentas de suporte à navegação - botões *back* e *forward*, histórico e *bookmarks*.

  Os botões *back* e *forward* (Figura 3.1b) possibilitam ao usuário, a partir do nó atual, retornar a um nó anterior ou seguir a um nó adiante, respectivamente. Para isso, os nós em questão devem ter sido anteriormente visitados.

  O histórico é um mecanismo que lista todos os nós previamente visitados e permite aos usuários o acesso direto a qualquer um desses nós.

  O *bookmarks* é um mecanismo semelhante ao histórico, mas os nós apresentados na lista são selecionados pelo usuário. Na Figura 3.1c é apresentado um exemplo de *bookmarks*. Comparada à lista do histórico, a do *bookmark* é menor e mais gerenciável, embora também possa não incluir todos os nós relevantes [Nielsen, 1995; Xu et al., 2001].

Considerando que a maioria dos espaços de informação são bem grandes, não é possível recuperar as informações manualmente, como ocorre por meio da navegação tradicional [Brusilovsky, 1996]. Nesse sentido, a navegação adaptativa pode auxiliar os usuários nessa busca, contemplando-os com alguns recursos.

3.2.2 Navegação Adaptativa

A proposta da navegação adaptativa é auxiliar os usuários através da adaptação na apresentação dos *links*. A informação é disponibilizada de maneira seletiva e contextual, considerando os diferentes objetivos e níveis de conhecimento dos usuários [Brusilovsky, 2001].

A maneira de apresentar os *links* permite distinguir cinco tipos de tecnologias de navegação adaptativa: orientação direta, classificação, ocultação, anotação e adaptação de mapas. Para comparar cada uma delas é necessário entender, inicialmente, "como" e "em qual contexto" os *links* são normalmente apresentados.

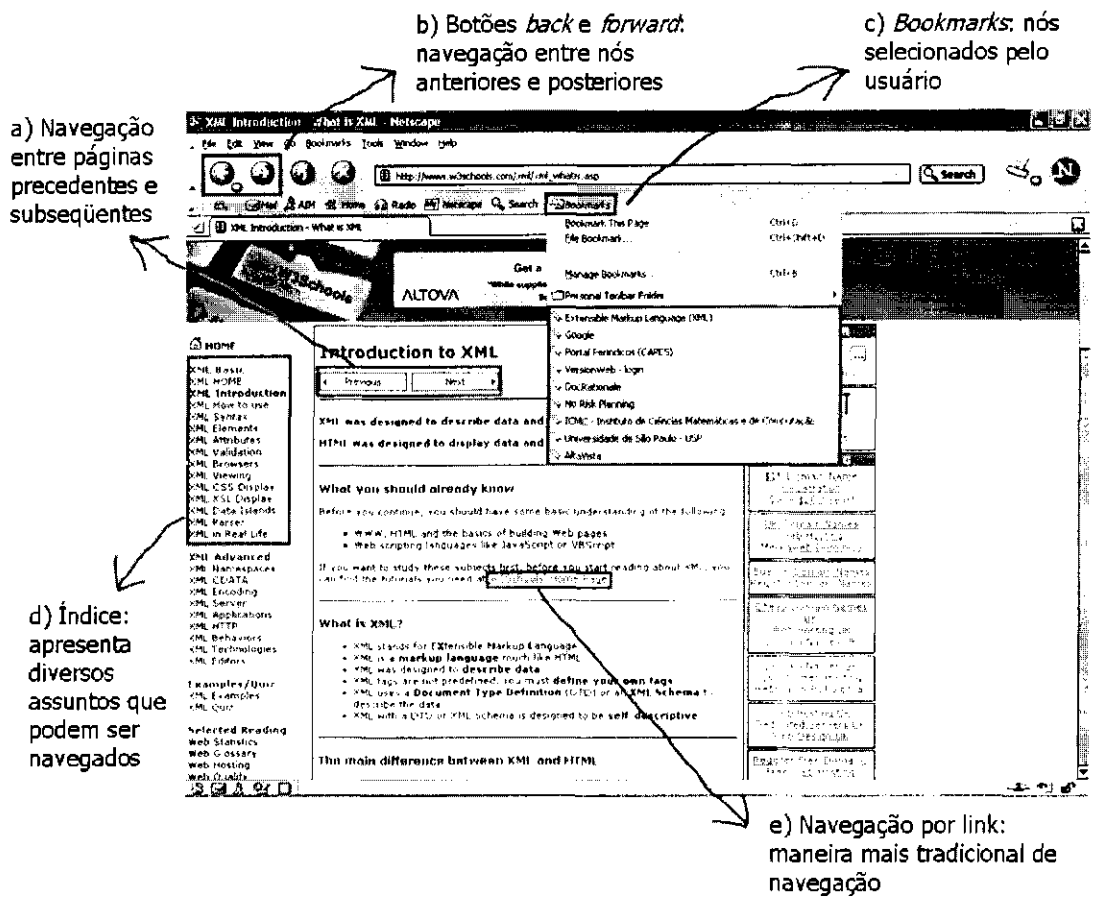

Figura 3.1: Exemplos de tipos de navegação tradicional

Do ponto de vista do que pode ser alterado e adaptado, os *links* são classificados em quatro tipos [Brusilovsky, 1996]:

1. *Links* locais e não contextuais: são *links* independentes do conteúdo apresentado na página. Podem ser, por exemplo, botões ou um menu *pop-up*. São facilmente manipulados.

2. *Links* contextuais: estão embutidos no conteúdo da página e não podem ser removidos. Podem ser, por exemplo, palavras e frases no texto.

3. *Links* de índices: a página de índices pode ser considerada um tipo especial de página que possui somente *links*. Freqüentemente, esses *links* são não contextuais e são apresentados em ordem alfabética na apresentação do conteúdo.

4. *Links* de mapas globais e locais: mapas globais e locais são representações gráficas do espaço total de informação ou de uma parte local desse espaço, respectivamente. Os

mapas são apresentados como uma rede de nós conectados por setas. Os usuários podem navegar diretamente sobre todos os nós visíveis no mapa.

Tendo visto a classificação dos *links*, torna-se possível entender as tecnologias de adaptação de *links* [Brusilovsky, 1996]:

- Orientação direta - Em cada ponto, qual a melhor direção a seguir?

  É a tecnologia mais simples de navegação adaptativa. A partir de um nó, deve-se decidir qual o melhor próximo nó a ser visitado pelo usuário, de acordo com seus objetivos. Além disso, a orientação direta é uma tecnologia clara e fácil de ser implementada, podendo ser utilizada com os quatro tipos de *links* mencionados. No entanto, o problema com essa tecnologia é que ela não oferece auxílio ao usuário que não seguir a sugestão dos nós.

- Classificação - Em que ordem os *links* devem ser apresentados?

  Essa tecnologia tem como proposta classificar os *links* de uma página *Web* de acordo com algum critério útil ao usuário. Um exemplo de critério seria: "os primeiros *links* apresentados em uma página são os *links* mais relevantes". A classificação pode ser utilizada com *links* não contextuais, dificilmente com *links* de índices, e não pode ser utilizada com *links* contextuais e de mapas.

- Ocultação - Que *links* não devem ser apresentados?

  A idéia da tecnologia de ocultação é restringir o espaço de navegação, escondendo os *links* de hiperdocumentos não relevantes. A ocultação protege o usuário da complexidade de um grande espaço de informações, é mais transparente e, além disso, pode ser utilizada com os quatro tipos de *links*.

- Anotação - Como agregar mais informação aos *links*?

  O propósito dessa tecnologia é acrescentar ao *link* algum tipo de comentário que possa dar ao usuário mais informações a respeito desse *link*. A anotação relacionada a um *link* pode ser estática ou dinâmica (estar associada a cada usuário). Assim como a ocultação, a tecnologia de anotação também pode ser utilizada com os quatro tipos de *links*.

- Adaptação de mapas - Como apresentar mapas?

  A tecnologia de adaptação de mapas abrange diversas maneiras de adaptar a apresentação dos mapas, globais e locais, apresentados ao usuário.

Todas as tecnologias de navegação adaptativa apresentadas são consideradas tecnologias primárias. A grande maioria das técnicas de navegação adaptativa utiliza exatamente uma dessas tecnologias. Além disso, essas tecnologias não são contraditórias e podem ser utilizadas de maneira combinada.

Embora a navegação caracterize o principal acesso às páginas *Web* [Halasz, 2001], outras tecnologias de recuperação de informação podem ser utilizadas. Dentre elas, as máquinas de busca (*search engines*) são ultimamente as mais utilizadas para a recuperação de informações na *Web*.

3.2.3 Máquinas de Busca

A *Web* revolucionou o acesso à informação não só pela enorme quantidade de informação disponível, mas também pela crescente eficiência para acessar tal informação. Além disso, a explosão na quantidade de informação propiciou o desenvolvimento de diversas tecnologias de recuperação.

Todas as máquinas de busca executam a mesma função: buscar e indexar páginas *Web*. Isto é, uma máquina de busca gerencia um índice na *Web* e apresenta aos usuários as páginas relevantes associadas à busca realizada [Kobayashi & Takeda, 2000].

Embora as máquinas de busca sejam implementadas de diferentes maneiras, existem algumas questões quanto à recuperação de informação que devem ser consideraradas no desenvolvimento dessas máquinas [Croft, 1995; Lawrence & Giles, 1998; Hu et al., 2001]:

- Relevância dos resultados da busca: os usuários das máquinas de busca na *Web* consideram a capacidade de buscar e identificar somente as páginas realmente relevantes ao tópico consultado, uma das principais características que essas máquinas devem possuir. No entanto, determinar quais páginas são relevantes o suficiente para serem recuperadas é um grande desafio.

- Acesso a informação atual: a natureza dinâmica dos documentos e das páginas na *Web* contribui para que parte dos documentos esteja desatualizada quando atingem o público alvo. Assim como na questão dos resultados relevantes, os usuários também esperam obter resultados atualizados.

- Recuperação de informação distribuída: os principais problemas são localizar as melho-

res bases de dados em um ambiente distribuído, e agrupar os resultados provenientes dessa busca distribuída. Os resultados devem ser agrupados de maneira a produzir uma ordenação completa dos itens recuperados, ao invés de uma coleção de ordenações individuais.

- Adaptação às consultas dos usuários: freqüentemente, as palavras usadas pelos usuários nas consultas são diferentes das encontradas nos documentos relevantes. Assim, técnicas que tratem esse problema expandindo automaticamente a consulta, procurando, por exemplo, por palavras diferentes com o mesmo significado, são altamente desejáveis.

- Tempo de resposta e atraso na comunicação: o tempo de resposta de uma consulta, juntamente com o atraso na comunicação, têm sido citados como os problemas mais comuns das máquinas de busca. *Links* "quebrados" também são citados como um problema freqüente.

- Interface e Navegação: as interfaces devem oferecer uma variedade de funções, incluindo formulação de consulta (características como habilidade de busca por frases, uso de termos de lógica booleana, habilidade de refinamento da consulta), apresentação da informação recuperada, *feedback* e navegação. O desafio é apresentar essas funcionalidades sofisticadas de maneira conceitualmente simples, tornando o sistema fácil de entender e utilizar.

As máquinas de busca na *Web*, apesar das diferentes implementações, possuem como estrutura básica a combinação entre busca, indexação (*indexing*) e agrupamento (*clustering*), consulta e ordenação (*ranking*) [Trevor et al., 2001], como apresentado na Figura 3.2.

Para a busca de páginas *Web* é utilizada, geralmente, uma aplicação que percorre automaticamente vários *Web sites*, recuperando as páginas. As aplicações também percorrem os *links* apontados por essas páginas a fim de encontrar outras páginas relevantes [Hu et al., 2001]. Essas aplicações são, em sua maioria, agentes inteligentes, conhecidos como *robots*, *spiders* ou *crawlers* [Trevor et al., 2001]. Como o crescimento das páginas na *Web* é grande, novas buscas são periodicamente realizadas para procurar por alterações. As páginas recuperadas pela busca são então indexadas.

A indexação (*indexing*) consiste em apresentar uma coleção de *links* de páginas *Web* [Kobayashi & Takeda, 2000]. Essa coleção compõe a base de dados de índices de uma máquina

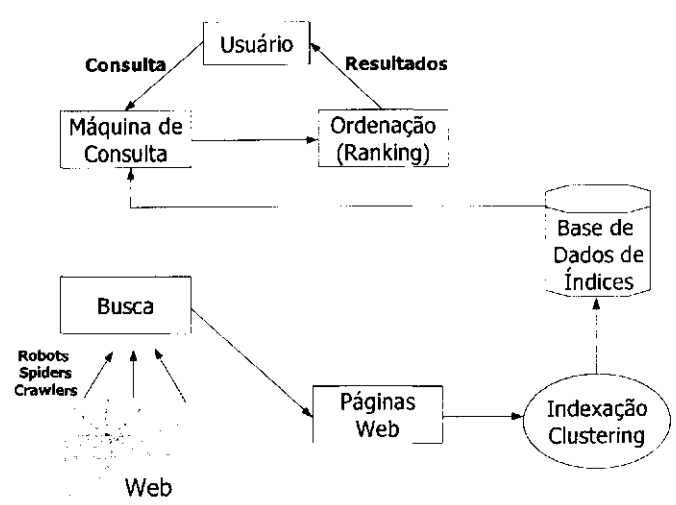

Figura 3.2: Estrutura básica de uma máquina de busca

de busca, que é a base utilizada para retornar os resultados de uma consulta feita pelo usuário. Por isso, na indexação, a qualidade das páginas referentes aos *links* da coleção também deve ser considerada [Henzinger et al., 1999]. Nesse contexto, um passo importante é identificar dois tipos importantes de páginas *Web*: *authorities* e *hubs*. *Authorities* são páginas que provêem as melhores fontes de informação sobre o tópico considerado. *Hubs* são páginas que apresentam uma lista de *links* de páginas *authorities* [Kobayashi & Takeda, 2000; Hu et al., 2001]. Assim, *links* de *authorities* e *hubs* devem estar sempre presentes na coleção de indexação. A grande quantidade de páginas e a freqüência em que crescem e são atualizadas, tornam a indexação de páginas *Web* uma tarefa aparentemente impossível [Kobayashi & Takeda, 2000].

Além da indexação, o *clustering* também é utilizado com o objetivo de melhorar a qualidade dos resultados retornados em uma consulta. *Clustering* consiste em agrupar documentos similares. Existem duas classes de medidas de semelhança: a semelhança de uma página em relação à consulta realizada e a semelhança entre duas páginas [Kobayashi & Takeda, 2000]. Assim, a utilização de *clustering* possibilita que os *links* de páginas similares sejam agrupados na coleção.

O processo de consulta é a atividade que analisa a consulta e a compara com os índices a fim de encontrar páginas relevantes ao tópico consultado [Hu et al., 2001]. Segundo Lidsky et al. [1997], as consultas dos usuários são divididas em cinco categorias:

1. Consulta simples: consiste na consulta por palavras ou frase. A consulta é limitada a elementos de uma página, como título ou URL, a um domínio específico ou a um *Web*

site.

2. Consulta personalizada: a partir dos resultados obtidos em uma consulta simples, pode-se refinar a consulta de maneira que os melhores resultados sejam o alvo da nova consulta.

3. Consulta em diretórios: trata-se de consultas em diretórios, restringindo-a com o objetivo de proporcionar melhor qualidade dos resultados.

4. Consulta por notícias atuais: consiste na busca por conteúdos recentes. Considera-se, normalmente, a data da informação recuperada.

5. Conteúdo da *Web*: nessa categoria de consulta, todos os conteúdos adicionais são considerados. Por exemplo, salas de bate papo, avisos e propagandas.

Nas máquinas de busca, a ordenação (*ranking*) das páginas *Web*, isto é, a ordem em que as páginas relevantes devem ser apresentadas ao usuário, é realizada com o auxílio de algoritmos implementados especialmente para esse propósito [Hu et al., 2001].

De maneira geral, pode-se dizer que a maioria das máquinas de busca atuais são baseadas em texto e apresentam os resultados de uma consulta em uma lista de *links*, que podem conter ou não o resumo das páginas recuperadas [Kobayashi & Takeda, 2000].

Com o advento de tantas máquinas de busca, o usuário tem que estar atento às características de cada uma delas, de maneira que ele consiga escolher corretamente uma máquina de busca que atenda suas necessidades. Assim, o usuário precisa saber como a máquina funciona, o que ela foi projetada para recuperar, onde está localizada, e até mesmo que tal máquina existe [Dreilinger & Howe, 1997].

Alguns exemplos de máquinas de busca encontradas atualmente na *Web* são: **Google** (`http://www.google.com`), **Alta Vista** (`http://www.altavista.com`), **Excite** (`http://www.excite.com`), **HotBot** (`http://www.hotbot.com`) e **Yahoo!** (`http://www.yahoo.com`).

As máquinas de busca propiciaram o desenvolvimento das *metasearches*. Essa tecnologia de recuperação de informação é apresentada na próxima seção.

3.2.4 Metasearches

Metasearches são máquinas que, automaticamente e simultaneamente, consultam diversas máquinas de busca, interpretam os resultados, e os apresentam em um formato integrado [Dreilinger & Howe, 1997]. As principais vantagens das *metasearches* são combinar os resultados de várias máquinas de busca e prover uma interface com o usuário consistente, de maneira que eles possam realizar buscas nessas máquinas [Lawrence & Giles, 1999]. Em outras palavras, as *metasearches* permitem que os usuários verifiquem que máquina de busca retorna os melhores resultados, sem que eles tenham que buscar em todas elas individualmente [Hu et al., 2001]. Apesar dessas vantagens, as *metasearches* podem introduzir suas próprias deficiências: (i) dificuldade em ordenar a lista de resultados, (ii) limitar o número de resultados obtidos, (iii) não oferecer suporte a todas características de cada máquina de busca [Lawrence & Giles, 1999]. Assim sendo, o sucesso das *metasearches* depende da seleção cuidadosa de quais recursos utilizar [Dreilinger & Howe, 1997].

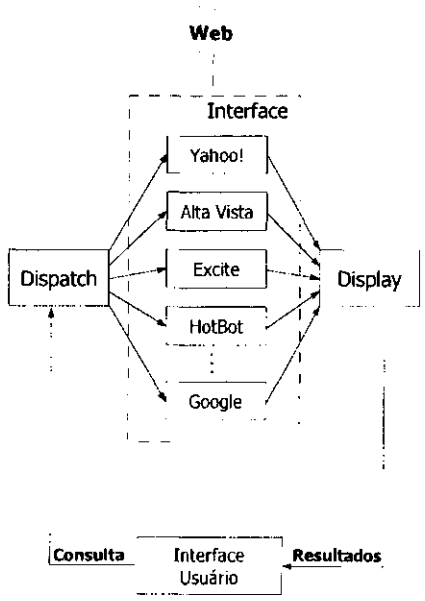

Figura 3.3: Estrutura básica de uma *metasearch*

A estrutura de uma *metasearch*, apresentada na Figura 3.3, é formada basicamente por três principais componentes: *dispatch*, *interface* e *display* [Dreilinger & Howe, 1997; Hu et al., 2001].

1. *Dispatch*: é o algoritmo ou abordagem de tomada de decisão. Determina a quais máquinas de busca uma consulta específica será enviada.

2. *Interface*: são programas que gerenciam a interação com uma determinada máquina de busca. Adaptam a consulta do usuário para o formato de uma máquina de busca em particular, e também são responsáveis por interpretar os diferentes formatos dos resultados retornados dessas máquinas de busca.

3. *Display*: os resultados de cada máquina de busca devem ser integrados para serem apresentados ao usuário. Os resultados podem ser apresentados em ordem ou intercalados. Possíveis resultados duplicados e *links* "quebrados" podem ser removidos.

Resumidamente, o *dispatch* determina a que máquinas de busca remotas enviar a consulta. Simultaneamente, a interface envia a consulta para as máquinas de busca selecionadas. Os resultados retornados são convertidos em um formato interno pela *interface*. O *display* integra os resultados, remove os resultados duplicados, e os apresenta ao usuário [Dreilinger & Howe, 1997].

Alguns exemplos de máquinas de *metasearch* são: **Discover** [Sheldon et al., 1995], **MetaCrawler** [Selberg & Etzioni, 1997], **Savvy Search** [Dreilinger & Howe, 1997] e **GlOSS** [Gravano et al., 1999].

Embora as máquinas de busca e as *metasearches* sejam as tecnologias de recuperação de informação mais utilizadas na *Web* nos dias de hoje, existem outras tecnologias referentes a esse assunto.

3.2.5 Outras Tecnologias de Recuperação de Informação na Web

Abordagens SQL

Abordagens de linguagem de consultas estruturadas - SQL (*Structured Query Language*) - enxergam a *Web* como um imenso banco de dados (em que cada registro seria uma página *Web*) e utiliza linguagens SQL-*like* para dar suporte efetivo e flexível ao processo de busca [Hu et al., 2001].

Exemplo de uma consulta em linguagem SQL-like:

Encontrar páginas da USP que tenham a palavra-chave "pós-graduação".

```
SELECT url FROM http"//*.usp.br/*
WHERE keyword LIKE 'pós-graduação';
```

onde "*" substitui qualquer seqüência de caracteres.

Algumas abordagens que utilizam linguagens SQL-*like* para realizar buscas na *Web* são: **W3QL** [Konopnicki & Shmueli, 1995], **WebLog** [Lakshmanan et al., 1996] e **WebSQL** [Mendelzon et al., 1997].

Recuperação Multimídia

O advento da multimídia acrescentou áudio, gráficos, imagens, vídeo e outros tipos de dados à *Web*. A dificuldade de realizar a indexação automática dos dados multimídia é um dos principais empecilhos ao desenvolvimento de máquinas de busca multimídia [Hu et al., 2001]. Algumas máquinas de busca de imagens e vídeos foram desenvolvidas, por exemplo, **VisualSEEk** [Smith & Chang, 1996], **QBIC** [Flickner et al., 1997] e **Yahoo! Picture Gallery** (`http://gallery.yahoo.com/`).

A recuperação de informação de imagens e vídeos são mais comuns. Por outro lado, a recuperação de informação de áudio é considerada um dos mais difíceis desafios para a busca multimídia, sendo poucas máquinas de busca desenvolvidas.

Recuperação em Documentos XML

Atualmente, existe a tendência de utilizar XML (*Extensible Markup Language*) como padrão de formato de documentos na *Web*. Assim, ampliam-se as pesquisas sobre recuperação de informações em documentos XML, especialmente porque para pesquisas baseadas em conteúdo, XML apresenta duas vantagens importantes [Fuhr, 2003]:

1. Devido ao fato de XML representar a estrutura lógica de um documento de maneira explícita, o sistema de recuperação pode retornar unidades lógicas como respostas às buscas.

2. Buscas mais precisas podem ser utilizadas, baseadas no conteúdo apresentado dentro de elementos específicos.

Algumas abordagens para recuperação de informação em documentos XML definem linguagens de consulta (*query languages*) para documentos XML, por exemplo, **XIRQL** [Fuhr & Grojohann, 2001] e **XML-GL** [Ceri et al., 1999]. Outras abordagens realizam o *clustering* de documentos XML de maneira a facilitar a recuperação dos mesmos [Lian et al., 2004; Yoon

et al., 2001]. Há ainda um sistema de busca de documentos XML que oferece suporte tanto a consultas baseadas na estrutura quanto no conteúdo desses documentos [Chang, 2001].

Até esse momento, apenas as tecnologias de recuperação na *Web* foram vistas. Como o mecanismo proposto neste trabalho visa recuperar DR, faz-se necessário abordar mais especificamente as abordagens de recuperação de DR.

3.3 Recuperação de DR

O acesso a DR possui diversos propósitos, dentre eles: responder a perguntas do usuário, mostrar aspectos importantes de determinado tópico, acompanhar o processo de desenvolvimento de projetos, obter documentos de artefatos de projeto [Gruber & Russell, 1992]. Assim, a recuperação de informações deve ser cuidadosamente considerada nos sistemas de DR.

Geralmente, a recuperação de DR é determinada pela representação utilizada na captura do DR. Dada uma representação, ferramentas que oferecem suporte, por exemplo, à navegação ou a outras estratégias de recuperação, podem ser implementadas [Regli et al., 2000]. Assim, a integração de sistemas de DR com outros sistemas que ofereçam suporte à recuperação de informação, pode melhorar bastante a qualidade da recuperação nos sistemas de DR [Regli et al., 2000].

Regli et al. [2000] apontam quatro abordagens básicas para a recuperação que podem ser utilizadas nos sistemas de DR: navegação, uso de *triggers*, recuperação baseada em consulta e estratégias de recuperação híbridas. Cada uma dessas abordagens será apresentada a seguir.

3.3.1 Navegação

Nos sistemas que utilizam a navegação como forma de recuperação, é permitido aos usuários navegar pelos nós de informação através dos *links*. Alguns exemplos de sistemas de DR que utilizam esse tipo de abordagem são:

- VIEWPOINTS [Fischer et al., 1991]: esse sistema cria um ambiente de projeto integrado para explorar as opções de projeto. É utilizado para procurar respostas relacionadas a problemas específicos.
- ADD [Fischer & McCall, 1989]: dispõe de uma interface *read-only* que permite aos

usuários navegar graficamente através das decisões e das justificativas relacionadas aos artefatos.

- COMET [Mark et al., 1992]: oferece ao usuário a possibilidade de examinar as descrições de módulos existentes que sejam relacionadas com as descrições de um novo módulo. Para auxiliar o usuário na navegação, o sistema apresenta uma janela com os *links* dos módulos.

- IDIS [Chung & Goodwin, 1998]: oferece facilidades de navegação a partir de diagramas do AutoCAD. Ao selecionar um item do diagrama, a especificação do mesmo é mostrada. Dessa maneira, os usuários podem encontrar a informação desejada sobre o item.

A grande desvantagem dessa abordagem está relacionada à quantidade de informação. Para artefatos complexos, grande quantidade de informação é armazenada, e encontrar informações específicas pode se tornar uma tarefa difícil [Regli et al., 2000].

3.3.2 Uso de Triggers

Alguns sistemas de DR utilizam *triggers* para monitorar a captura de DR. O processo de projeto é examinado e as decisões tomadas são comparadas considerando algumas restrições, regras ou critérios. Se diferenças são detectadas, o DR é automaticamente recuperado [Regli et al., 2000].

Sistemas que utilizam esse tipo de abordagem de recuperação são:

- PHIDIAS [Shipman & McCall, 1997]: o objetivo principal do sistema é conectar o DR com a tarefa de projeto dos artefatos. O DR é conectado ao desenho do artefato e a algumas operações específicas do artefato projetado. Existem ainda as críticas que são conectadas à tarefa de projeto do artefato. Tais críticas podem ser executadas pelos próprios usuários ou "disparadas" automaticamente dependendo das condições de projeto.

- CRACK [Fischer & McCall, 1989]: esse sistema oferece suporte baseado em conhecimento. As críticas são apoiadas por sistemas de suporte inteligente que detectam e criticam as soluções parciais realizadas pelo usuário, baseados no conhecimento de princípios de projeto. Uma crítica pode ser "disparada" ao examinar a base de conhecimento e ser detectado que a decisão tomada não é satisfatória.

- ADD+ [Garcia & Souza, 1997]: esse sistema age como um aprendiz. O sistema deve aprender as características que tornam um caso diferente do padrão no processo de projeto. Assim, ele deve ser capaz de acessar o conhecimento de projeto, propiciando a justificativa para esse novo caso.

A proposta dos sistemas que utilizam *triggers* é facilitar a recuperação de DR, uma vez que a busca de informações é realizada de maneira automática.

3.3.3 Recuperação Baseada em Consulta

Em geral, a recuperação de DR em sistemas que utilizam a abordagem de recuperação baseada em consulta é mais eficiente que a recuperação em sistemas que utilizam a navegação [Regli et al., 2000]. As consultas são questões que podem ser respondidas através da exploração de diversas opções ou a partir do *backtracking* na rede de nós e *links*.

Os sistemas a seguir utilizam a abordagem de recuperação baseada em consulta:

- REMAP/MM [Ramesh & Sengupta, 1995]: utiliza uma linguagem de consulta dedutiva que define diferentes tipos de consultas *ad hoc* e oferece uma interface gráfica para mostrar as consultas e recuperar as informações desejadas. Esse sistema utiliza multimídia e combinação de representação formal e informal na captura de DR.

- KRITIK [Chandrasekaran & Iwasaki, 1993]: esse sistema utiliza um modelo estrutura-comportamento-função (*structure-behavior-function* - SBF) para explicar como a estrutura de um dispositivo executa sua função. O objetivo do modelo é atender um dos propósitos de DR que é mostrar "como" e "porque" os projetistas conseguem que os dispositivos funcionem como planejado.

- DRIVE [Garza & Alcantara, 1997]: é um sistema baseado em regras, no qual o DR capturado é estruturado e pode ser processado automaticamente pelo sistema. Uma base de regras é utilizada para examinar a validade das novas relações toda vez que um parâmetro de projeto muda. Essa base de regras também é utilizada para detectar conflitos e oferecer aos usuários diferentes opções para a solução.

De maneira geral, as explicações de DR são geradas a partir de informações relevantes capturadas durante o projeto e de conhecimento pré-adquirido. Entretanto, prover uma metodologia para selecionar o conhecimento é um dos problemas desses sistemas [Regli et al., 2000].

3.3.4 Estratégias Híbridas de Recuperação

As estratégias híbridas de recuperação podem combinar quaisquer abordagens de recuperação apresentadas anteriormente. Os sistemas JANUS e KBDS-IBIS utilizam estratégias híbridas.

- JANUS [Fischer & McCall, 1989]: utiliza os sistema de críticas do CRACK, mas permite aos usuários também adicionar críticas relacionadas à tarefa em questão. Esse sistema também oferece apresentação *online* e permite que os usuários naveguem pela base de conhecimento.

- KBDS-IBIS [Banares-Alcantara & King, 1997]: esse sistema oferece diferentes meios de utilizar o DR. Por exemplo, avaliação automática das posições e geração automática de relatórios. Uma vez que o sistema requer informações precisas para determinar a validade dos argumentos armazenados, pode-se considerar que o KBDS-IBIS não é apenas um sistema para recuperação do DR armazenado, pois outras informações devem ser inferidas [Regli et al., 2000].

O resumo dos sistemas de DR e suas respectivas abordagens de recuperação é apresentado na Tabela 3.1.

Tabela 3.1: Alguns sistemas de DR e suas abordagens de recuperação

Sistema de DR	**Modelo de Representação**	**Abordagem de Recuperação**
ADD	Argumentação e Baseado em Modelo	Navegação
ADD+	Estrutura Retórica	Triggers
COMET	Loom	Navegação
CRACK	Não Disponível	Triggers
DRIVE	PDN	Baseada em Consulta
IDIS	IBIS	Navegação
JANUS	PHI	Recuperação Híbrida
KBDS-IBIS	IBIS	Recuperação Híbrida
KRITIK	SBF	Baseada em Consulta
PHIDIAS	PHI	Triggers
REMAN/MM	IBIS	Baseada em Consulta
VIEWPOINTS	PHI	Navegação

Nos sistemas de DR encontrados na literatura, observa-se que a quantidade é praticamente a mesma para as diferentes abordagens de recuperação de DR.

3.4 Considerações Finais

A recuperação de informação é extremamente importante nos dias atuais. Para os usuários, encontrar informações relevantes sobre um tópico desejado, é um dos principais pontos considerados quando se trata de recuperar informação.

Neste capítulo foram apresentadas diferentes tecnologias de recuperação de informação na *Web*. O desenvolvimento dessas tecnologias foi motivado tanto pela chegada da *Web* assim como pela explosão das páginas nela localizadas. Navegação, máquinas de busca, *metasearchengines*, dentre outras, foram as tecnologias apresentadas.

De maneira semelhante às tecnologias de recuperação de informação na *Web*, as abordagens adotadas nos sistemas de DR para busca de informações também foram destacadas. As abordagens mencionadas, em relação aos sistemas de DR, foram: navegação, uso de *triggers*, recuperação baseada em consulta e estratégias híbridas.

A DocRationale, uma ferramenta *Web* para captura e recuperação de DR utiliza a abordagem de navegação para a recuperação do DR armazenado. Como de trata de importante objeto de estudo neste trabalho, a ferramenta DocRationale é apresentada no próximo capítulo.

CAPÍTULO

4

A Ferramenta DocRationale

4.1 Considerações Iniciais

Busca-se capturar e recuperar o DR dos projetos de maneira automatizada através do desenvolvimento de sistemas de DR apoiados por computador. Na literatura, diversos sistemas de DR têm sido propostos. As características de captura e armazenamento são determinadas pela perspectiva sobre DR adotada. Isto é, se um sistema possui a captura facilitada, sua recuperação, em geral, é mais complexa e vice-versa. O ideal é que os sistemas de DR possibilitem a captura com exigência mínima de esforço dos usuários, proporcionando ao mesmo tempo, uma recuperação efetiva.

A necessidade de armazenar informações de projeto, tanto registradas nos diversos documentos, quanto as relacionadas a DR, e possibilitar sua efetiva disponibilização sem exigir tanto esforço dos usuários motivou o desenvolvimento da ferramenta DocRationale. A ferramenta utiliza uma combinação das perspectivas de argumentação e comunicação. Também são utilizadas anotações que, na ferramenta, são estruturadas de maneira a refletir o modelo de representação simplificado, adotado na DocRationale.

Em relação à DocRationale, são apresentadas, neste capítulo, sua descrição, arquitetura, funcionamento, uma nova funcionalidade adicionada, análise de uso e novas idéias para aper-

feiçoar as funcionalidades da ferramenta. A recuperação de DR na DocRationale também é descrita neste capítulo.

4.2 Descrição

A DocRationale [Francisco et al., 2003; Francisco, 2004] é uma ferramenta *Web*, implementada para permitir a aquisição, estruturação, armazenamento e recuperação de DR relacionado a artefatos de software (todos os tipos de documentos de software, sejam na forma de diagramas, textos ou de outras mídias). Além do DR, dados sobre os projetos, como título, fases e atividades envolvidas, artefatos, também são registrados na ferramenta.

A DocRationale foi desenvolvida utilizando-se um servidor *Web* estendido com um interpretador PHP (*Hypertext Preprocessor*) e o banco de dados relacional MySQL. A ferramenta oferece *login* remoto e acesso controlado às informações armazenadas. Para isso, no entanto, é necessário que o usuário seja cadastrado na ferramenta. Foram definidos na DocRationale quatro tipos de usuários:

1. Administrador de usuários - é responsável por cadastrar usuários, autorizar ou rejeitar uma solicitação de cadastro, e definir em qual tipo esses usuários serão cadastrados na ferramenta.

2. Administrador de projetos - é responsável por gerenciar projetos, artefatos e questões. Também pode navegar sobre eles.

3. Membro de projetos - esse usuário pode inserir artefatos e questões, e navegar sobre projetos, artefatos e questões.

4. Visitante - a esse usuário só é permitido navegar sobre projetos, artefatos e questões.

Diferentes equipes podem ser definidas para os diferentes projetos inseridos na DocRationale. Assim, um usuário que é **Membro** em um projeto, pode ser apenas **Visitante** em um outro projeto. O fato da DocRationale ser uma ferramenta *Web*, permite que os membros de uma equipe estejam fisicamente distantes, mas possam trabalhar juntos no desenvolvimento do projeto, ou seja, a DocRationale privilegia o fator de colaboração entre os membros das equipes de projeto para obtenção de DR.

Para a aquisição e o armazenamento de DR, foi adotada na DocRationale uma combinação das perspectivas de comunicação e argumentação. Sob a perspectiva de comunicação, a DocRationale possibilita que diversas formas de comunicação digital (arquivos de áudio, vídeo, e-mail entre outros) sejam anexadas, complementando as informações sobre os artefatos de software. Quanto à perspectiva de argumentação, foram utilizadas características do modelo PHI para estruturar as informações de maneira hierárquica. Na ferramenta, o modelo foi simplificado para utilizar apenas três tipos de entidades: **Questões**, **Posições** e **Argumentos**. Assim, o DR relacionado aos artefatos é determinado por um conjunto de Questões, Posições e Argumentos, obtidos na forma de anotações (consideradas um bom meio para aquisição de DR [Souza et al., 1999]).

Com o propósito de apoiar a aquisição e recuperação de DR, houve a integração da DocRationale com CoTeia [Arruda et al., 2002] e GroupNote [Izeki et al., 2001] como pode ser visto na Figura 4.1. CoTeia é uma ferramenta hipermídia colaborativa assíncrona de criação de páginas *Web*. Possui diversas funcionalidades, sendo as mais interessantes para a DocRationale: (i) criação e edição de páginas *Web*, que são feitas no próprio *browser* via formulário HTML (*HyperText Markup Language*); (ii) histórico, que permite o acesso ao conteúdo das últimas versões de cada hiperdocumento; e (iii) *upload*, que possibilita a submissão de arquivos de qualquer formato ao servidor CoTeia. GroupNote é um serviço aberto de anotações colaborativas na *Web* implementado como uma API (*Application Programming Interface*). Suas características mais interessantes para DocRationale são: (i) compartilhamento de anotações por equipes de usuários; e (ii) fornecimento da funcionalidade de hierarquia de nós.

A criação e atualização das páginas dos projetos na CoTeia são de responsabilidade da DocRationale. Existem funções definidas na DocRationale que transmitem seus dados de modo a serem compreendidos por outras funções definidas na CoTeia e GroupNote. Assim, a cada inserção de um projeto na base de dados da DocRationale, a CoTeia também é atualizada e um *link* para a página correspondente na CoTeia é inserido na DocRationale. De maneira semelhante, a cada inserção de uma questão na base de dados da DocRationale, uma anotação é criada no GroupNote. Em outras palavras, a CoTeia auxilia a criação das páginas dos projetos (disponibilizadas na própria DocRationale) e o GroupNote auxilia a aquisição de DR através de anotações hierarquizadas representadas no modelo PHI simplificado adotado na ferramenta.

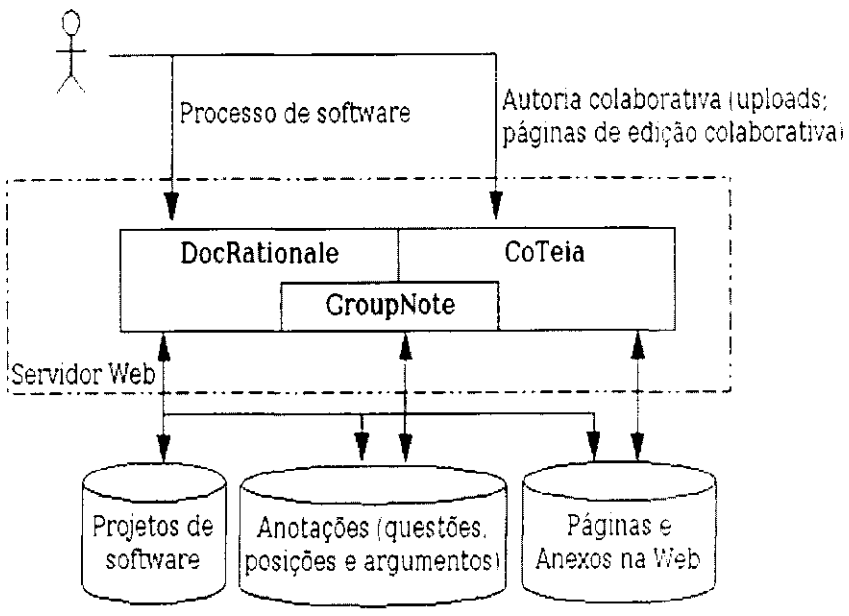

Figura 4.1: Visão geral da integração da DocRationale com CoTeia e GroupNote [Francisco, 2004]

4.3 Arquitetura

Os dados dos projetos, assim como os DRs relacionados aos artefatos de software, são armazenados em bases de dados relacionais, distribuídas entre a DocRationale, a CoTeia e o GroupNote como apresentado na Figura 4.2. Nas bases de dados da DocRationale são armazenados os dados sobre os projetos: fases, atividades, artefatos e questões. As páginas dos projetos e os arquivos anexos são armazenados na base de dados da CoTeia e na base de dados do GroupNote são armazenados os DRs dos artefatos.

A DocRationale, como pode ser visto também na Figura 4.2, possui diversos módulos específicos que juntos desempenham todas as funcionalidades disponíveis na ferramenta. O Controle de Acesso (CA) é o responsável por controlar o acesso aos outros módulos da DocRationale, CoTeia e GroupNote. O módulo Gerenciador de Usuários (GU) só é acessado por usuários do tipo **Administradores de Usuários**. Os módulos gerenciadores de Projetos (GP), Fases (GF), Atividades (GAt), Artefatos (GAr), Questões (GQ) e Equipe (GE) são responsáveis pela inserção, edição e exclusão dos seus respectivos dados. Os módulos GP, GF, GAt, GAr, GQ e GE apresentam uma interdependência. Os módulos Tradutor de Dados para CoTeia (TDC) e Tradutor de Dados para GroupNote (TDG) são responsáveis pela comunicação entre a

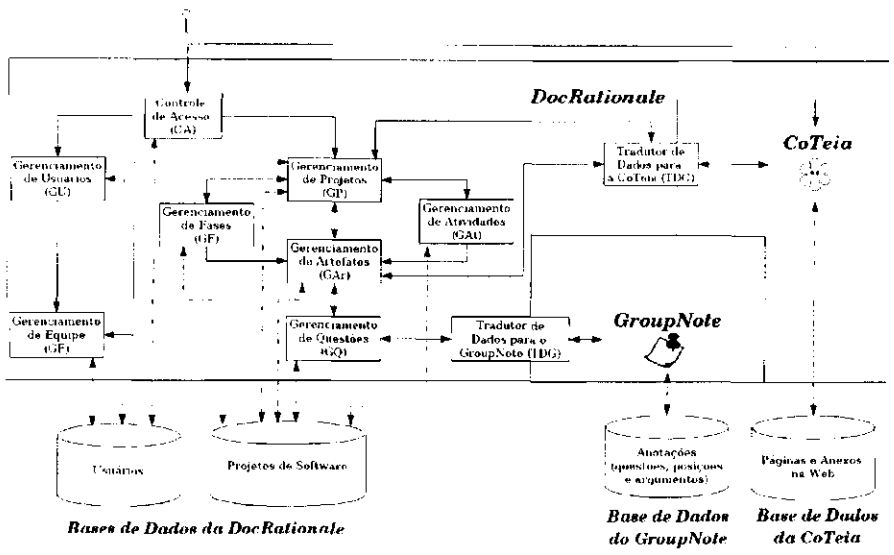

Figura 4.2: Arquitetura da DocRationale [Francisco, 2004]

DocRationale e a CoTeia e GroupNote, respectivamente.

4.4 Apresentação das Informações na DocRationale

A apresentação de projetos da DocRationale, por meio da tela mostrada na Figura 4.3, é a primeira visão de um **Administrador de Projetos** após o seu *login*. Na Figura 4.3a, tem-se uma lista de projetos inseridos na ferramenta. As funcionalidades que podem ser executadas em relação aos projetos estão destacadas na Figura 4.3b.

Outras funcionalidades relacionadas mais diretamente à ferramenta como os botões de modelos disponíveis, ajuda, alterar a senha e sair do sistema são mostradas na Figura 4.3c. O símbolo da CoTeia, destacado na Figura 4.3d, é o *link* para a página do respectivo projeto na CoTeia. Na figura 4.3e é apresentada a legenda que identifica o *status* dos projetos.

A interface da DocRationale é composta por diversas outras telas, sendo que as telas de Artefatos e Questões são bem similares à tela da Figura 4.3. Um exemplo de página de projeto gerada pela DocRationale pode ser visto na Figura 4.4. As informações destacadas na Figura 4.4a são os dados gerais sobre o projeto, por exemplo, título, data de início, data prevista para o término. Logo abaixo dessas informações (Figura 4.3b) são apresentados os *links* dos artefatos envolvidos no projeto em questão. Ao clicar sobre um desses *links*, é mostrada a página gerada para o artefato correspondente.

Na Figura 4.5, pode-se ver um exemplo de página de artefato. De maneira semelhante à

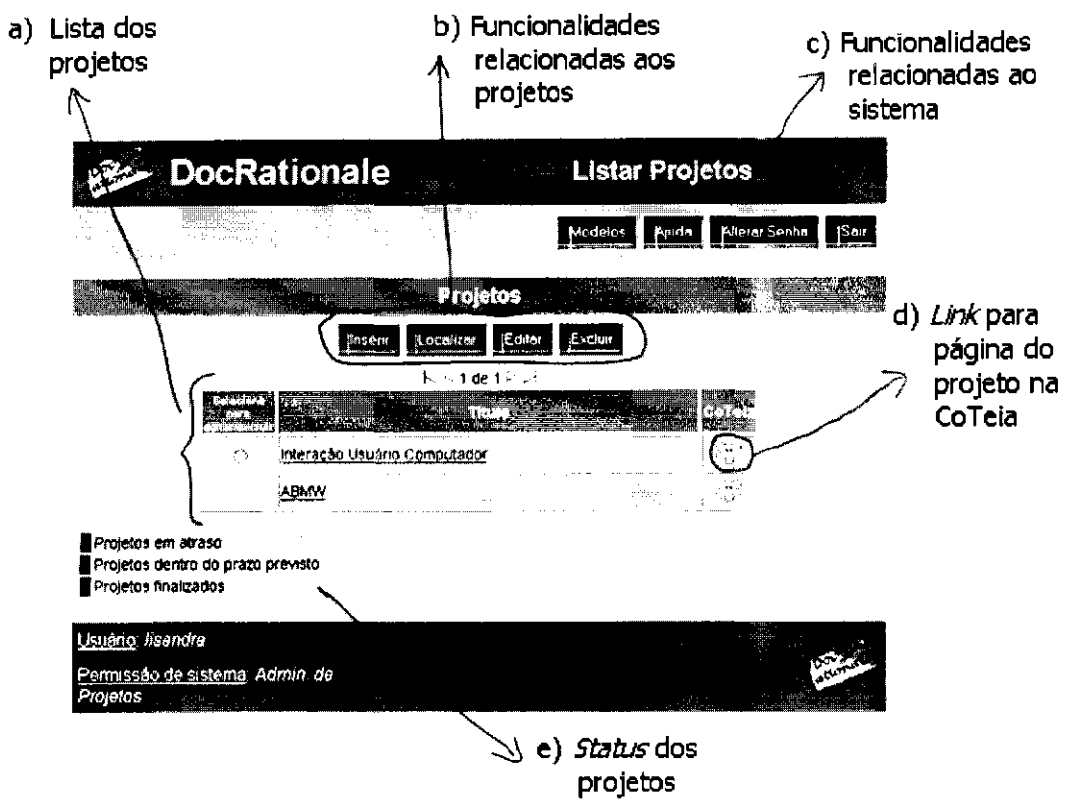

Figura 4.3: Tela que lista os projetos da DocRationale

página de projeto, são mostradas informações gerais sobre o artefato (Figura 4.5a) e logo abaixo, a lista de todas as questões envolvidas com esse artefato. A partir dessa página é efetuado o *upload* dos arquivos anexos. O *link* para essa funcionalidade é destacado na Figura 4.5b. Na Figura 4.5c são mostrados os *links* de todas as Questões relacionadas a esse Artefato. Clicando sobre esses *links* é possível ver as Posições e os Argumentos da referida questão. Os *links* dos arquivos anexados, relacionados ao artefato, são exibidos na Figura 4.5d. Ao clicar sobre esses *links*, pode-se realizar o *download* do arquivo em questão.

Com o objetivo de analisar o uso da ferramenta DocRationale, uma pesquisa foi realizada com alunos de graduação que a utilizaram para gerenciar e capturar o DR dos projetos realizados em determinada disciplina. A análise dos resultados obtidos é a apresentada a seguir.

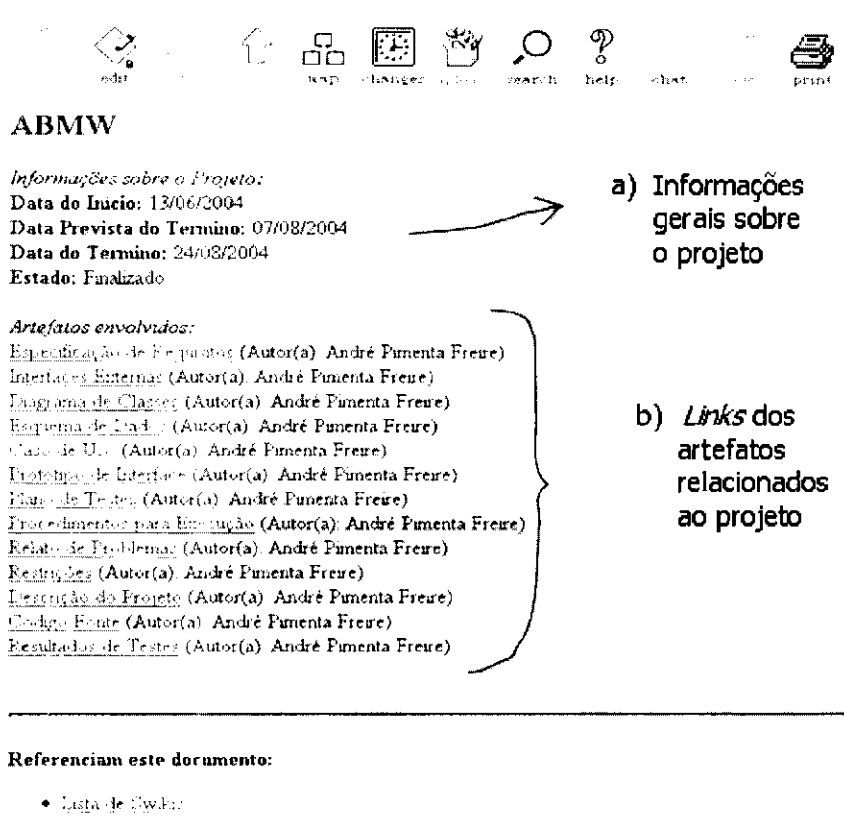

Figura 4.4: Página de um projeto na CoTeia

4.5 Análise de Uso da DocRationale

Para realizar a pesquisa sobre a utilização da DocRationale, elaborou-se um questionário (vide Apêndice A) cujo propósito foi coletar informações relacionadas, especialmente, à facilidade de compreensão e de uso da ferramenta. Alunos de graduação em computação cursando a disciplina de Hipermídia durante o segundo semestre de 2003 utilizaram a ferramenta e responderam ao questionário. A análise das respostas mostrou que inicialmente, a ferramenta não era facilmente compreendida e que as funcionalidades demoravam a ser descobertas.

Outro estudo sobre a usabilidade da DocRationale, também realizado com alunos de graduação, mas que cursavam a disciplina de Interação Usuário-Computador, mostrou diversas deficiências em relação a esse item na ferramenta. Foi constatado que para executar as funcionalidades oferecidas pela DocRationale, o usuário deve realizar muitos cliques, ou seja, muitos *links* e algumas telas são apresentadas ao usuário desnecessariamente. Outro problema destacado é que muitas informações são exibidas em uma mesma tela, deixando muitas vezes o

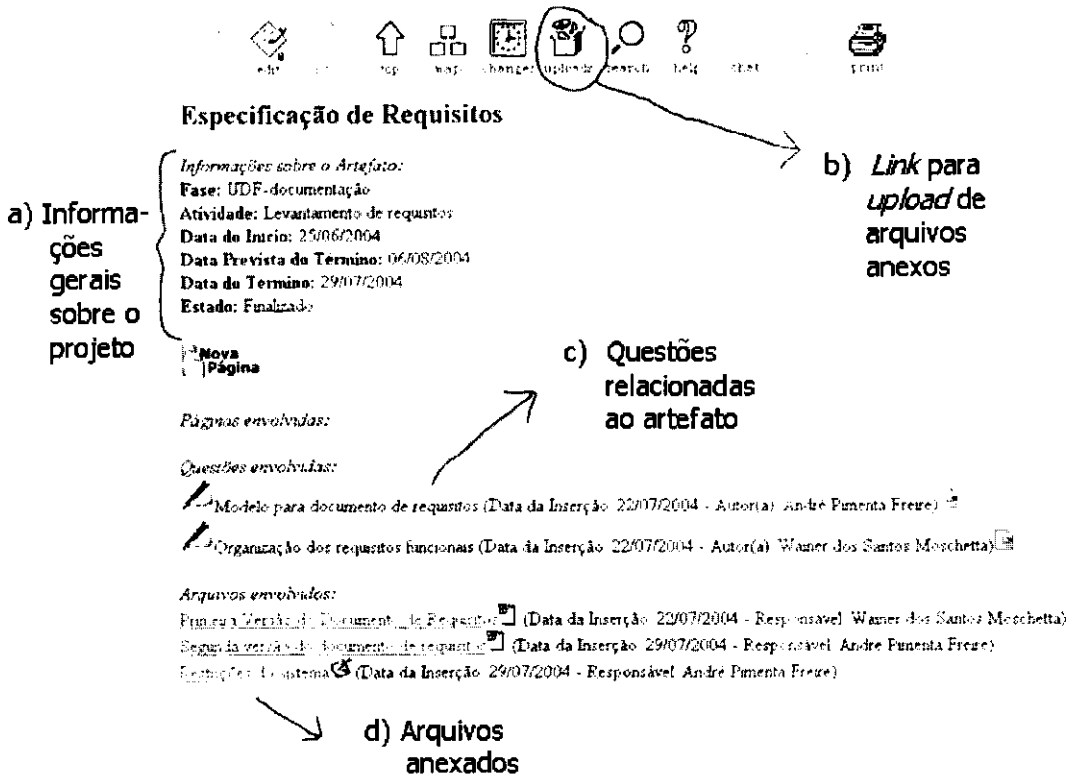

Figura 4.5: Página de um artefato na Coteia

usuário confuso. O fato de algumas palavras apresentarem sentido vago também foi um ponto salientado. Também foi evidenciado que a ferramenta pode deixar os usuários perdidos. Nesse sentido, uma solução para orientar os usuários seria utilizar um histórico de navegação, mostrando ao usuário de onde ele veio e onde ele está.

Devido a essas diversas deficiências, algumas melhorias têm sido propostas. Uma solução mais imediata com o objetivo de auxiliar o entendimento da DocRationale e de transpor os obstáculos iniciais para o uso da ferramenta, foi a elaboração e disponibilização de documentos de ajuda na DocRationale.

Esses documentos, ou melhor, hiperdocumentos foram editados com a ferramenta XML-mind Editor. Para elaborar tais hiperdocumentos foi realizada uma análise exaustiva de toda a interface provida pela DocRationale. Uma breve descrição da ferramenta e de suas funcionalidades é apresentada nesses hiperdocumentos. Deve-se ressaltar que existem três documentos diferentes, uma para cada tipo de usuário. Assim, a um usuário **Membro de Projeto** não são exibidas as funcionalidades que só se referem a um usuário **Administrador de Projetos**, por

exemplo.

Na Figura 4.6 é mostrado o documento de ajuda para o usuário do tipo **Administrador de Projetos**. Esse documento é o mais extenso. A descrição da DocRationale é apresentada através dos itens **Introdução**, **Estrutura**, **Acesso** e **Funcionalidade do Sistema**. Esses itens são os mesmos em todos os documentos. As funcionalidades relacionadas a projetos, artefatos e questões são descritas nos itens correspondentes. Para cada uma dessas funcionalidades, são mostradas no hiperdocumento as telas exibidas pela própria ferramenta. Tal recurso tem o propósito de mostrar passo a passo a execução de cada funcionalidade.

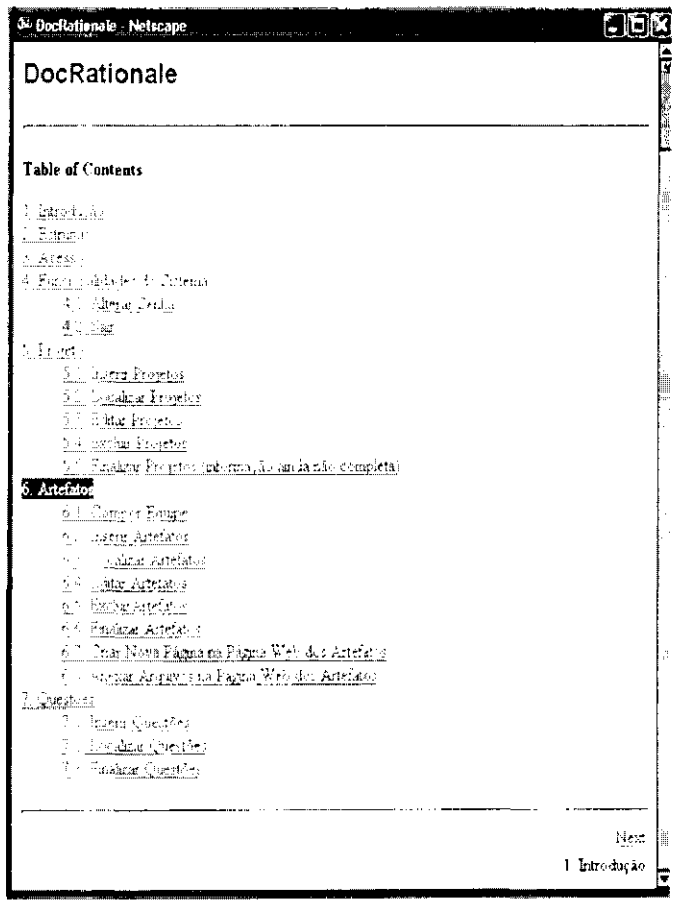

Figura 4.6: Documento de ajuda disponível na DocRationale

Além dos documentos de ajuda, também foi implementada na DocRationale uma funcionalidade para facilitar a inserção de projetos, reduzindo o número de passos para efetuar essa ação. Essa nova funcionalidade é detalhada na próxima seção.

4.6 Inserção Automática de Projetos via Documento XML

Uma funcionalidade, adicionada à DocRationale mais recentemente, permite que um projeto de software seja criado na ferramenta a partir de um documento XML.

XML (*Extensible Markup Language*), como o próprio nome já diz, é uma linguagem de marcação extensível que tem a finalidade de descrever informações. A grande vantagem é ser possível criar linguagens de marcação personalizadas. A XML também oferece uma abordagem para descrição, captura, processamento e publicação de informações [W3C, 2001a; McGrath, 1999]. Devido a esses e outros benefícios, os documentos XML foram adotados como integrantes de soluções para novas funcionalidades na ferramenta DocRationale. Maiores detalhes sobre a linguagem XML e os documentos XML serão dados no Capítulo 5.

O objetivo dessa nova funcionalidade foi atender a demanda didática de uso da ferramenta uma vez que, ao ser solicitado um projeto a diversos alunos (ou grupos), pode-se definir previamente o modelo de processo do projeto a ser desenvolvido. O documento XML utilizado para a inserção de projetos, cuja estrutura tem que estar de acordo com um DTD (*Document Type Definition*) previamente definido, deve conter dados sobre o projeto como fases, atividades, artefatos e suas respectivas datas de início e de previsão de término. Para a definição do DTD foi necessário analisar quais os dados obrigatórios para a criação de um novo projeto na DocRationale. Assim, o DTD reflete a exigência desses dados, mas também possibilita a criação de um projeto completo, não restringindo, por exemplo, o número de fases, atividades e artefatos no projeto.

O Gerente de Projeto pode definir seu próprio modelo de documento XML ou utilizar, caso existam, modelos disponíveis na própria DocRationale. A DocRationale carrega esse documento XML, insere automaticamente os dados nele contidos na base de dados da ferramenta e cria um novo projeto.

Na Figura 4.7 é apresentada a tela da DocRationale a partir da qual é possível acessar o DTD e os modelos, caso existam modelos disponíveis. Os modelos de projeto disponíveis para *download* são mostrados na Figura 4.7a. Na Figura 4.7b é apresentado o DTD.

Um exemplo de documento XML é apresentado na Figura 4.8. Através desse exemplo é possível saber que o projeto em questão possui as fases **Definição**, **Desenvolvimento** e **Finalização**. Além disso, também pode-se notar que, por exemplo, a fase de **Desenvolvimento** é

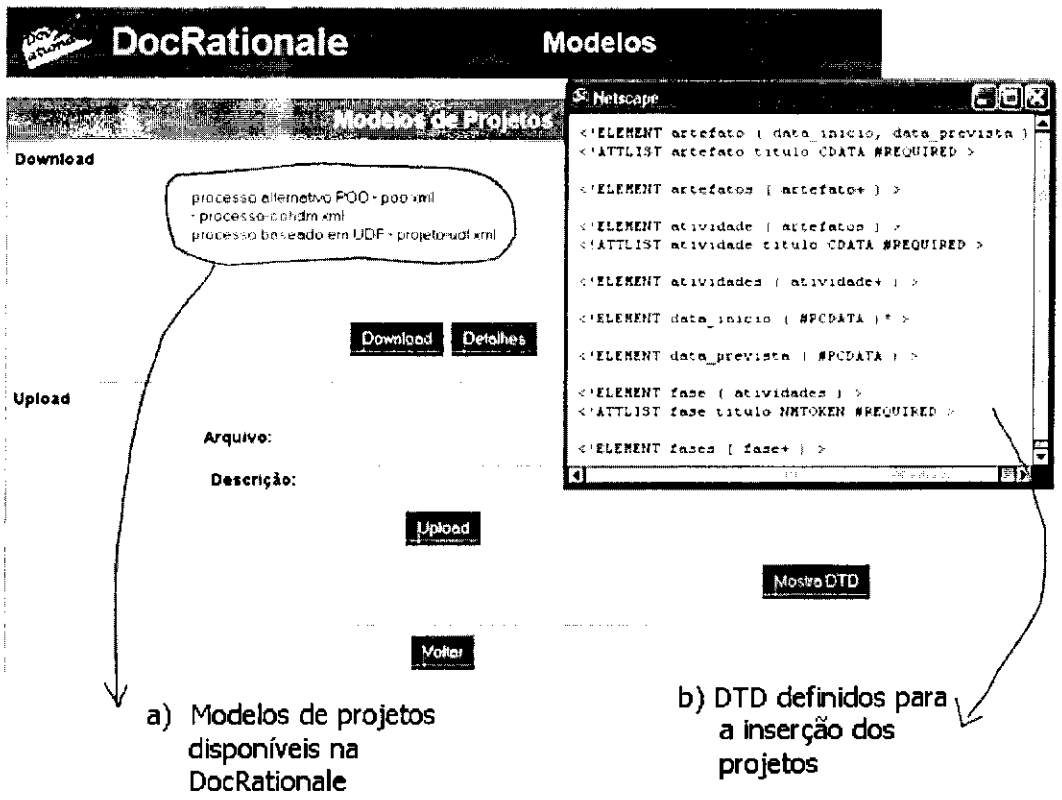

Figura 4.7: DTD e relação de modelos disponíveis na DocRationale

constituída das atividades de **Planejamento** e **Elaboração**. Por sua vez, a atividade de **Planejamento** possui os artefatos: **Requisitos**, **Interface externa** e **Descrição do projeto**, e a atividade **Elaboração** possui os artefatos **Código**, **Plano de Teste**, **Resultado do Teste**, **Procedimentos de Construção**, **Relatório de Problemas**, **Anotações** e **Revisão**.

A definição do DTD utilizado nessa nova funcionalidade, assim como a elaboração dos hiperdocumentos de ajuda, foram contribuições à DocRationale realizadas como parte deste trabalho.

Outra melhoria desejável para a DocRationale é a implementação de uma nova funcionalidade que ofereça outros modelos de representação para estruturar o DR capturado. Essa funcionalidade é descrita a seguir.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE fases SYSTEM "projeto1.dtd">
  <fases>
   <fase titulo="Definição">
    <atividades>
     <atividade titulo="Especificação do
                         trabalho">
      <artefatos>
       <artefato titulo="Enunciado">
        <data_inicio>03/05/2004
        </data_inicio>
        <data_prevista>10/05/2004
        pt</data_prevista>
       </artefato>
      </artefatos>
     </atividade>
    </atividades>
   </fase>
   <fase titulo="Desenvolvimento">
    <atividades>
     <atividade titulo="Planejamento">
      <artefatos>
       <artefato titulo="Requisitos">
        <data_inicio>03/05/2004</data_inicio>
        <data_prevista>10/05/2004
        </data_prevista>
       </artefato>
       <artefato titulo="Interface externa">
        <data_inicio>10/05/2004</data_inicio>
        <data_prevista>17/05/2004
        </data_prevista>
       </artefato>
       <artefato titulo="Descrição do
                         projeto">
        <data_inicio>17/05/2004</data_inicio>
        <data_prevista>24/05/2004
        </data_prevista>
       </artefato>
      </artefatos>
     </atividade>
     <atividade titulo="Elaboração">
      <artefatos>
       <artefato titulo="Código">
        <data_inicio>24/05/2004</data_inicio>
        <data_prevista>07/06/2004
        </data_prevista>
       </artefato>
       <artefato titulo="Plano de Teste">
        <data_inicio>07/06/2004</data_inicio>
        <data_prevista>14/06/2004
        </data_prevista>
       </artefato>
       <artefato titulo="Resultado do Teste">
        <data_inicio>14/05/2004</data_inicio>
        <data_prevista>18/05/2004
        </data_prevista>
       </artefato>
       <artefato titulo="Procedimentos de
                         Construção">
        <data_inicio>24/05/2004</data_inicio>
        <data_prevista>07/06/2004
        </data_prevista>
       </artefato>
       <artefato titulo="Relatório de
                         Problemas">
        <data_inicio>24/05/2004</data_inicio>
        <data_prevista>18/06/2004
        </data_prevista>
       </artefato>
       <artefato titulo="Anotações">
        <data_inicio>24/05/2004</data_inicio>
        <data_prevista>18/06/2004
        </data_prevista>
       </artefato>
       <artefato titulo="Revisão">
        <data_inicio>18/06/2004</data_inicio>
        <data_prevista>25/06/2004
        </data_prevista>
       </artefato>
      </artefatos>
     </atividade>
    </atividades>
   </fase>
   <fase titulo="Finalização">
    <atividades>
     <atividade titulo="Verificação">
      <artefatos>
       <artefato titulo="Avaliação do
                         trabalho">
        <data_inicio>25/06/2004</data_inicio>
        <data_prevista>30/06/2004
        </data_prevista>
       </artefato>
      </artefatos>
     </atividade>
    </atividades>
   </fase>
  </fases>
```

Figura 4.8: Exemplo de documento XML para um projeto

4.7 Aperfeiçoamento na Representação de DR

Como já foi visto anteriormente, na DocRationale é utilizado o modelo PHI simplificado para representar o DR capturado. Pretende-se com essa nova funcionalidade a ser implementada, ampliar as formas de representação de DR para a captura na ferramenta. Além de prover outras opções, essa funcionalidade facilitaria a captura dos DRs dos projetos, pois a aquisição seria realizada de maneira semelhante à inserção automática dos projetos, isto é, via documentos XML.

Os documentos XML foram adotados para definir, linearmente, os modelos de representação de DR mais comumente utilizados para a perspectiva de argumentação: IBIS, PHI, QOC e DRL. A partir de cada um dos modelos de representação de DR foram definidos os *XML Schema Definition* (XML Schema ou XSD) correspondentes (vide Apêndice B). Para a definição dos XML Schemas, foram consideradas as entidades, suas relações e hierarquia. Cada XML Schema é o documento que descreve seu respectivo documento XML.

As particularidades de cada XML definido são mostradas nas próximas subseções. Além disso, todos os exemplos também apresentados se referem a um mesmo processo de tomada de decisão, representados linearmente nos modelos IBIS, PHI, QOC e DRL. A decisão é onde armazenar documentos XML gerados para os projetos existentes na DocRationale.

4.7.1 Representação Linear do IBIS

Na figura 4.9 o exemplo de DR é representado no modelo IBIS. Existe uma questão inicial e diferentes respostas a ela. Para cada resposta, são apresentados alguns argumentos contra ou a favor.

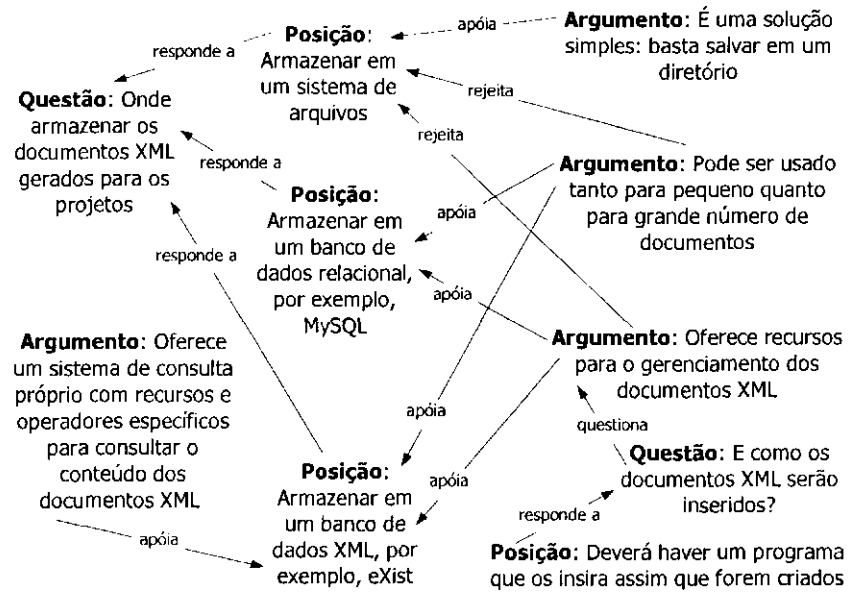

Figura 4.9: Exemplo de DR representado no modelo IBIS

Na figura 4.10, tem-se o documento XML que representa o exemplo apresentado anteriormente. Os elementos (**QUESTION**, **POSITION** e **ARGUMENT**) representam as entidades

do modelo de representação. O atributo **LINK** representa o relacionamento entre as entidades. Para representar o modelo na forma linear, foram necessárias algumas adaptações. Para a questão inicial, deve-se definir o LINK como "initial". Mesmos argumentos que se referem a diferentes posições devem ser repetidos.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<IBIS>
  <QUESTION LINK="initial">
      <QUESTION_TITLE> Onde armazenar os documentos XML gerados para os projetos
      </QUESTION_TITLE>
      <POSITION LINK="responde a">
          <POSITION_TITLE> Armazenar em um sistema de arquivos
          </POSITION_TITLE>
          <ARGUMENT LINK="apóia">
              <ARGUMENT_TITLE>
                    E uma solução simples:  basta salvar em um diretório
              </ARGUMENT_TITLE>
          </ARGUMENT>
          <ARGUMENT LINK="rejeita">
              <ARGUMENT_TITLE>
                    Oferece recursos para o gerenciamento dos documentos XML
              </ARGUMENT_TITLE>
          </ARGUMENT>
          <ARGUMENT LINK="rejeita">
              <ARGUMENT_TITLE>
                    Pode ser usado tanto para pequeno quanto para grande número de documentos
              </ARGUMENT_TITLE>
          </ARGUMENT>
      </POSITION>
      <POSITION LINK="responde a">
          <POSITION_TITLE> Armazenar em um banco de dados relacional, por exemplo, MySQL
          </POSITION_TITLE>
          <ARGUMENT LINK="apóia">
              <ARGUMENT_TITLE> Oferece recursos para o gerenciamento dos documentos XML
              </ARGUMENT_TITLE>
          </ARGUMENT>
          <ARGUMENT LINK="apóia">
              <ARGUMENT_TITLE>
                    Pode ser usado tanto para pequeno quanto para grande número de documentos
              </ARGUMENT_TITLE>
          </ARGUMENT>
      </POSITION>
      <POSITION LINK="responde a">
          <POSITION_TITLE> Armazenar em um banco de dados XML, por exemplo, eXist
          </POSITION_TITLE>
          <ARGUMENT LINK="apóia">
              <ARGUMENT_TITLE>
                    Pode ser usado tanto para pequeno quanto para grande número de documentos
```

```
            </ARGUMENT_TITLE>
        </ARGUMENT>
        <ARGUMENT LINK="apóia">
            <ARGUMENT_TITLE>
                Oferece recursos para o gerenciamento dos documentos XML
            </ARGUMENT_TITLE>
            <QUESTION LINK="questiona">
                <QUESTION_TITLE>
                    E como os documentos XML serão inseridos?
                </QUESTION_TITLE>
                <POSITION LINK="responde a">
                    <POSITION_TITLE> Deverá haver um programa que os insira assim que forem
                                criados
                    </POSITION_TITLE>
                </POSITION>
            </QUESTION>
        </ARGUMENT>
        <ARGUMENT LINK="apóia">
            <ARGUMENT_TITLE>
                Oferece um sistema de consulta próprio com recursos e operadores específicos
                para consultar o conteúdo dos documentos XML
            </ARGUMENT_TITLE>
        /ARGUMENT
    </POSITION>
  </QUESTION>
</IBIS>
```

Figura 4.10: Documento XML para representação linear do IBIS

4.7.2 Representação Linear do PHI

O exemplo representado no modelo PHI é apresentado na Figura 4.11. Assim como no modelo IBIS, há uma questão inicial para a qual são dadas diferentes respostas. Essas respostas, por sua vez, têm argumentos associados a elas.

O documento XML definido para representar o exemplo é mostrado na Figura 4.12. De maneira semelhante ao modelo IBIS, as entidades são representadas pelos elementos **ISSUE**, **SUB_ISSUE**, **ANSWER**, **SUB_ANSWER** e **ARGUMENT**.

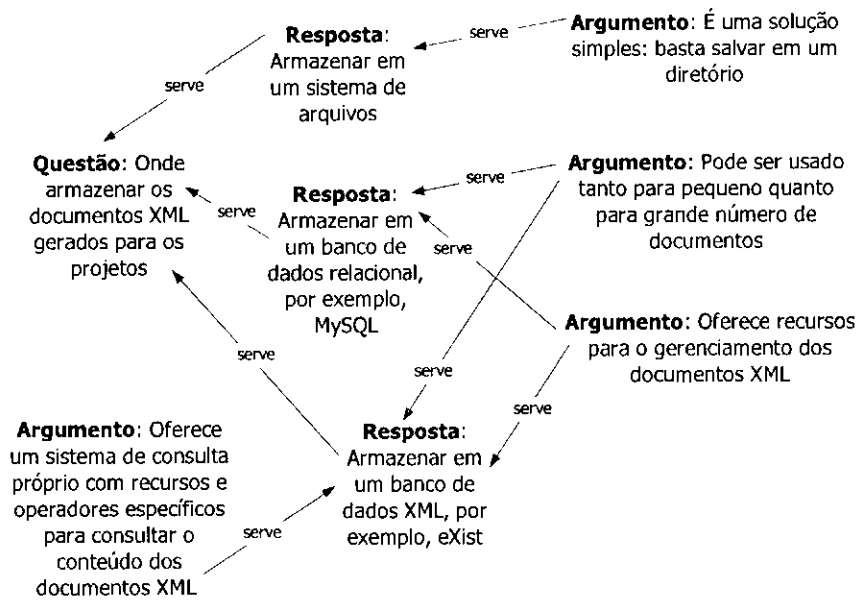

Figura 4.11: Exemplo de DR representado no modelo PHI

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<PHI>
  <ISSUE>
      <ISSUE_TITLE>
          Onde armazenar os documentos XML gerados para os projetos
      </ISSUE_TITLE>
      <ANSWER>
          <ANSWER_TITLE> Armazenar em um sistema de arquivos
          </ANSWER_TITLE>
          <ARGUMENT>
              <ARGUMENT_TITLE> É uma solução simples:  basta salvar em um diretório
              </ARGUMENT_TITLE>
          </ARGUMENT>
      </ANSWER>
      <ANSWER>
          <ANSWER_TITLE> Armazenar em um banco de dados relacional, por exemplo, MySQL
          </ANSWER_TITLE>
          <ARGUMENT>
              <ARGUMENT_TITLE>
                  Oferece recursos para o gerenciamento dos documentos XML
              </ARGUMENT_TITLE>
          </ARGUMENT>
          <ARGUMENT>
              <ARGUMENT_TITLE>
                  Pode ser usado tanto para pequeno quanto para grande número de documentos
              </ARGUMENT_TITLE>
          </ARGUMENT>
      </ANSWER>
```

```
<ANSWER>
    <ANSWER_TITLE> Armazenar em um banco de dados XML, por exemplo, eXist
    </ANSWER_TITLE>
    <ARGUMENT>
        <ARGUMENT_TITLE>
            Oferece recursos para o gerenciamento dos documentos XML
        </ARGUMENT_TITLE>
    </ARGUMENT>
    <ARGUMENT>
        <ARGUMENT_TITLE>
            Pode ser usado tanto para pequeno quanto para grande número de documentos
        </ARGUMENT_TITLE>
    </ARGUMENT>
    <ARGUMENT>
        <ARGUMENT_TITLE>
            Oferece um sistema de consulta próprio com recursos e operadores específicos
            para consultar o conteúdo dos documentos XML
        </ARGUMENT_TITLE>
    </ARGUMENT>
</ANSWER>
</ISSUE>
</PHI>
```

Figura 4.12: Documento XML para representação linear do PHI

4.7.3 Representação Linear do QOC

O exemplo representado no modelo QOC é mostrado na Figura 4.13. A uma questão inicial são apresentadas diferentes opções. Para cada uma das opções existem critérios, cujos relacionamentos são avaliados positiva ou negativamente. Existe ainda uma questão que surgiu como conseqüência de uma opção.

O documento XML referente a esse exemplo é mostrado na Figura 4.14. Esse documento possui os elementos **ISSUE**, **OPTION** e **CRITERION**, que representam as entidades do modelo de representação QOC.

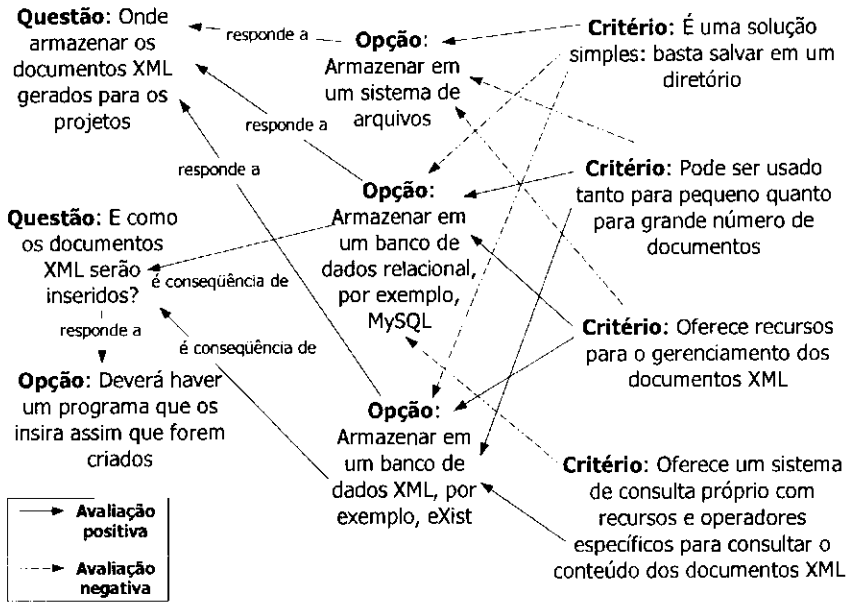

Figura 4.13: Exemplo de DR representado no modelo QOC

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<QOC>
  <QUESTION LINK="initial">
      <QUESTION_TITLE>
            Onde armazenar os documentos XML gerados para os projetos
      </QUESTION_TITLE>
      <OPTION LINK="responde a">
          <OPTION_TITLE> Armazenar em um sistema de arquivos
          </OPTION_TITLE>
          <CRITERION LINK="avaliação positiva">
              <CRITERION_TITLE>
                    É uma solução simples:  basta salvar em um diretório
              </CRITERION_TITLE>
          </CRITERION>
          <CRITERION LINK="avaliação negativa">
              <CRITERION_TITLE>
                    Pode ser usado tanto para pequeno quanto para grande número de documentos
              </CRITERION_TITLE>
        </CRITERION>
        <CRITERION LINK="avaliação negativa">
            <CRITERION_TITLE>
                  Oferece recursos para o gerenciamento dos documentos XML
            </CRITERION_TITLE>
        </CRITERION>
    </OPTION>
    <OPTION LINK="responde a">
        <OPTION_TITLE> Armazenar em um banco de dados relacional, por exemplo, MySQL
        </OPTION_TITLE>
        <CRITERION LINK="avaliação negativa">
```

```
            <CRITERION TITLE>
                  É uma solução simples:  basta salvar em um diretório
            </CRITERION_TITLE>
        </CRITERION>
        <CRITERION LINK="avaliação positiva">
            <CRITERION_TITLE>
                  Pode ser usado tanto para pequeno quanto para grande número de documentos
            </CRITERION_TITLE>
        </CRITERION>
        <CRITERION LINK="avaliação positiva">
            <CRITERION_TITLE>
                  Oferece recursos para o gerenciamento dos documentos XML
            </CRITERION_TITLE>
        </CRITERION>
        <CRITERION LINK="avaliação negativa">
            <CRITERION_TITLE>
                  Oferece um sistema de consulta próprio com recursos e operadores
                  específicos para consultar o conteúdo dos documentos XML
            </CRITERION_TITLE>
        </CRITERION>
        <QUESTION LINK="é conseqüência de">
            <QUESTION_TITLE> E como os documentos XML serão inseridos?
            </QUESTION_TITLE>
            <OPTION LINK="responde a">
                <OPTION_TITLE>
                    Deverá haver um programa que os insira assim que forem criados
                </OPTION_TITLE>
            </OPTION>
        </QUESTION>
    </OPTION>
    <OPTION LINK="responde a">
        <OPTION_TITLE> Armazenar em um banco de dados XML, por exemplo, eXist
        </OPTION_TITLE>
        <CRITERION LINK="avaliação negativa">
            <CRITERION_TITLE>
                  É uma solução simples:  basta salvar em um diretório
            </CRITERION_TITLE>
        </CRITERION>
        <CRITERION LINK="avaliação positiva">
            <CRITERION_TITLE>
                  Pode ser usado tanto para pequeno quanto para grande número de documentos
            </CRITERION_TITLE>
        </CRITERION>
        <CRITERION LINK="avaliação positiva">
            <CRITERION_TITLE>
                  Oferece recursos para o gerenciamento dos documentos XML
            </CRITERION_TITLE>
        </CRITERION>
```

```
        <CRITERION LINK="avaliação positiva">
            <CRITERION_TITLE>
                  Oferece um sistema de consulta próprio com recursos e operadores
                  específicos para consultar o conteúdo dos documentos XML
            </CRITERION_TITLE>
        </CRITERION>
        <QUESTION LINK="é conseqüência de">
            <QUESTION_TITLE> E como os documentos XML serão inseridos?
            </QUESTION_TITLE>
            <OPTION LINK="responde a">
                <OPTION_TITLE>
                    Deverá haver um programa que os insira assim que forem criados
                </OPTION_TITLE>
            </OPTION>
        </QUESTION>
    </OPTION>
  </QUESTION>
</QOC>
```

Figura 4.14: Documento XML para representação linear do QOC

4.7.4 Representação Linear do DRL

Na Figura 4.15 é mostrado o exemplo representado no modelo DRL. Inicialmente, o problema a ser decidido é apontado. Para esse problema existem alguns objetivos. As alternativas apresentadas procuram atender os objetivos. Alegações também são feitas em relação às alternativas.

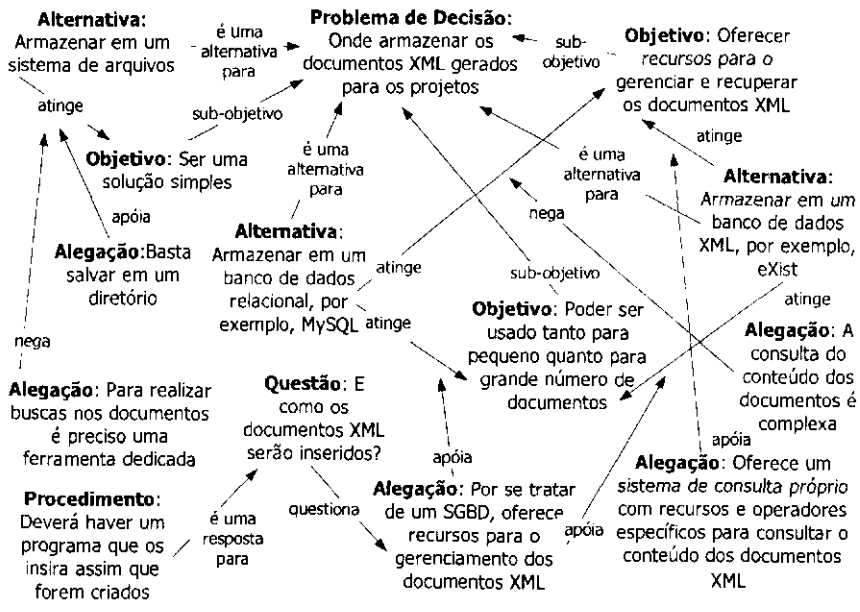

Figura 4.15: Exemplo de DR representado no modelo DRL

O documento XML definido para representar lineramente o exemplo é mostrado na Figura 4.16. As entidades são representadas pelos elementos **DECISON_PROBLEM**, **GOAL**, **ALTERNATIVE**, **CLAIM**, **QUESTION** e **PROCEDURE**.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DRL>
  <DECISION_PROBLEM LINK="initial">
      <DECISION_PROBLEM_TITLE>
          Onde armazenar os documentos gerados para os projetos
      </DECISION_PROBLEM_TITLE>
      <GOAL LINK="sub-objetivo">
          <GOAL_TITLE> Ser uma solução simples
          </GOAL TITLE>
          <ALTERNATIVE LINK="atinge">
              <ALTERNATIVE_TITLE>
                  Armazenar em um sistema de arquivo
              </ALTERNATIVE_TITLE>
              <CLAIM LINK="apóia">
                   <CLAIM_TITLE> Basta salvar em um diretório
                  </CLAIM_TITLE>
              </CLAIM>
              <CLAIM LINK="nega">
                   <CLAIM_TITLE>
                      Para realizar buscas nos documentos é preciso uma ferramenta dedicada
                  </CLAIM TITLE>
              </CLAIM>
          </ALTERNATIVE>
      </GOAL>
      <GOAL LINK="sub-objetivo">
          <GOAL_TITLE>
               Poder ser usado tanto para pequeno quanto para grande número de documentos
          </GOAL_TITLE>
          <ALTERNATIVE LINK="atinge">
              <ALTERNATIVE_TITLE>
                  Armazenar em um banco de dados relacional, por exemplo, MySQL
              </ALTERNATIVE_TITLE>
              <CLAIM LINK="apóia">
                  <CLAIM_TITLE>
                      Por se tratar de um SGBD, oferece recursos para o gerenciamemto dos
                      documentos XML
                  </CLAIM_TITLE>
                  <QUESTION LINK="questiona">
                  </CLAIM_TITLE>
                  <QUESTION LINK="questiona">
                      <QUESTION_TITLE> E como os documentos XML serão inseridos?
                      </QUESTION TITLE>
                      <PROCEDURE LINK="é uma resposta para">
                          <PROCEDURE_TITLE>
                              Deverá haver um programa que os insira assim que forem criados
                          </PROCEDURE_TITLE>
```

```
                    </PROCEDURE>
                </QUESTION>
            </CLAIM>
        </ALTERNATIVE>
        <ALTERNATIVE LINK-"atinge">
            <ALTERNATIVE_TITLE>
                Armazenar em um banco de dados XML, por exemplo, eXist
            </ALTERNATIVE TITLE>
            <CLAIM LINK="apóia">
                <CLAIM_TITLE>
                    Por se tratar de um SGBD, oferece recursos para o gerenciamemto dos
                    documentos XML
                </CLAIM_TITLE>
                <QUESTION LINK="questiona">
                    <QUESTION_TITLE> E como os documentos XML serão inseridos?
                    </QUESTION_TITLE>
                    <PROCEDURE LINK="é uma resposta para">
                        <PROCEDURE_TITLE>
                            Deverá haver um programa que os insira assim que forem criados
                        </PROCEDURE_TITLE>
                    </PROCEDURE>
                </QUESTION>
            </CLAIM>
        </ALTERNATIVE>
    </GOAL>
    <GOAL LINK="sub-objetivo">
        <GOAL_TITLE>
            Oferecer recursos para gerenciar e recuperar os documentos XML
        </GOAL_TITLE>
        <ALTERNATIVE LINK="atinge">
            <ALTERNATIVE_TITLE>
                Armazenar em um banco de dados relacional, por exemplo, MySQL
            </ALTERNATIVE_TITLE>
            <CLAIM LINK="nega">
                <CLAIM_TITLE>
                    A consulta do conteúdo dos documentos é complexa
                </CLAIM_TITLE>
            </CLAIM>
        </ALTERNATIVE>
        </CLAIM>
    </ALTERNATIVE>
    <ALTERNATIVE LINK="atinge">
        <ALTERNATIVE_TITLE>
            Armazenar em um banco de dados XML, por exemplo, eXist
        </ALTERNATIVE_TITLE>
        <CLAIM LINK="apóia">
            <CLAIM_TITLE>
                Oferece um sistema de consulta próprio com recursos e operadores
                específicos para consultar o conteúdo dos documentos XML
            </CLAIM_TITLE>
        </CLAIM>
```

```
            </ALTERNATIVE>
        </GOAL>
    </DECISION_PROBLEM>
</DRL>
```

Figura 4.16: Documento XML para representação linear do DRL

Nesta seção, os documentos XML definidos para representar os modelos IBIS, PHI, QOC e DRL foram apenas apresentados. Uma análise desses modelos é realizada a seguir.

4.7.5 Análise sobre as Representações Lineares

Nos documentos XML definidos para estruturar os modelos de representação na forma linear, existe um elemento que identifica o modelo (IBIS, PHI, QOC e DRL). Todos os outros elementos, exceto os da forma ***_TITLE**, representam as entidades. Estes últimos apresentam o conteúdo a elas relacionado. Os relacionamentos entre as entidades são representados pelo atributo **LINK**. O QOC não apresenta esse atributo, uma vez que para esse modelo existe apenas um tipo de relacionamento.

Apesar da estrutura dos documentos ser simples, algumas adaptações foram necessárias para que os modelos pudessem ser representados linearmente. Nos modelos IBIS e QOC, nos quais pode existir mais de uma entidade **Questão**, as questões que dão início a uma discussão devem ter o atributo definido como "initial". O mesmo acontece com o DRL em relação à entidade **Problema de Decisão**.

Na representação linear, uma mesma entidade que possui relacionamentos com outras entidades precisa ser repetida. Para atender essa exigência dos modelos, o XML Schema teve que ser definido para possibilitar essas repetições.

Uma particularidade da representação linear do modelo DRL é que apenas as alternativas que não tiverem relacionamentos com algum dos objetivos apresentarão, declaradamente, o LINK igual a "é uma alternativa para". As outras apresentarão o LINK igual a "atinge" e fica subentendido que ela é uma alternativa para o problema de decisão.

Quanto à complexidade, pode-se dizer que ela aumenta à medida que o número de entidades e relacionamentos dos modelos aumenta. A expressividade, especificidade e granularidade da representação do DR estão relacionadas diretamente ao modelo, sendo que a representação linear, exceto pelas características apresentadas anteriormente, não altera a representação.

Na ferramenta DocRationale, o processamento desses documentos seria realizado de maneira semelhante ao que acontece com os projetos. Uma vez criado o documento XML, e validado contra o XML Schema correspondente, os DRs seriam inseridos automaticamente.

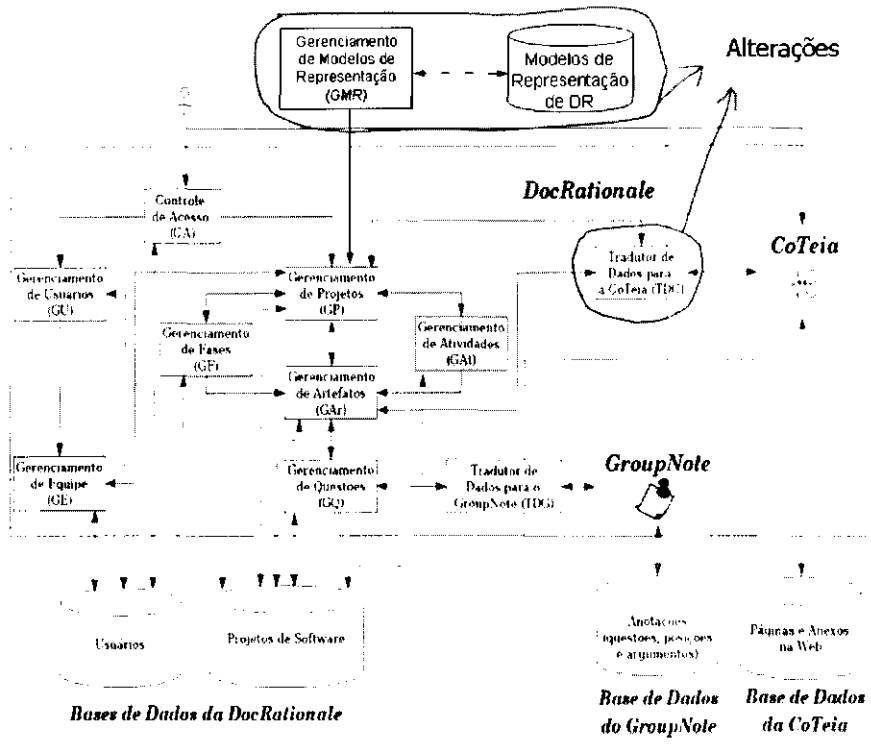

Figura 4.17: Módulo no qual seriam necessárias alterações

Considerando essa abordagem, a principal alteração na arquitetura da DocRationale seria criar uma nova base de dados e um módulo gerenciador dos modelos de representação. A base de dados armazenaria os dados provenientes dos documentos XML e o módulo gerenciador seria responsável pela inserção, edição e processamento dos dados dos documentos XML. Outra alteração seria modificar o módulo tradutor de dados para a CoTeia (TDC) para que esses novos dados também pudessem ser transmitidos da DocRationale para a CoTeia, que os disponibilizaria na página do projeto ao qual o DR está relacionado. Essa última alteração, entretanto, não modificaria a arquitetura da DocRationale.

Além de melhorias como a elaboração de um documento de ajuda, a inserção automática de projetos e o aperfeiçoamento da representação de DR, também pretende-se melhorar a recuperação de DR na DocRationale. Uma visão geral de como está a recuperação na ferramenta é apresentada a seguir.

4.8 Recuperação de DR na DocRationale

Atualmente, na ferramenta DocRationale é utilizada a abordagem de navegação através de *links*. A DocRationale possibilita ao usuário navegar através dos *links* dos projetos presentes na interface da própria ferramenta, ou a partir dos *links* apresentados nas páginas *Web* geradas, também pela ferramenta, para cada projeto.

Inúmeros projetos podem ser capturados e armazenados na DocRationale. Assim, a quantidade de informação de DR registrada pode tornar o espaço de informação bastante grande, dificultando a recuperação do DR através da navegação, pois o usuário deve percorrer grande quantidade de *links* para tentar encontrar as informações desejadas.

Visando solucionar esses problemas, foi definido um mecanismo que integra diferentes tecnologias *Web*. Esse mecanismo tem por objetivo promover a recuperação de DR através de consultas, que têm como ponto de partida palavras-chave que devem ser buscadas.

Existem alguns sistemas de DR que também utilizam consultas para a recuperação. A título de comparação com a DocRationale, esses sistemas são apresentados na próxima seção.

4.9 Sistemas de DR que oferecem Recuperação Baseada em Consulta

Dentre os sistemas de DR, citados na literatura, que possuem recuperação baseada em consulta tem-se: KRITIK, DRIVE e REMAP/MM. Infelizmente, a descrição de como a recuperação de DR ocorre é feita de maneira muito vaga, impedindo comparar de modo satisfatório a recuperação em tais sistemas com o mecanismo de recuperação de DR proposto neste trabalho para a ferramenta DocRationale.

O KRITIK [Chandrasekaran & Iwasaki, 1993] é um sistema *Case-Base*, ou seja, existe um processo através do qual novas situações em um projeto são resolvidas computacionalmente, a partir da recuperação e modificação de situações anteriores. Uma vez encontrada a solução para essa nova situação, essa solução se torna parte da base de conhecimento do sistema. Para esse sistema, diz-se apenas que a recuperação das situações é realizada através de consultas que têm como ponto inicial uma questão.

O DRIVE (*Design Rationale for the Information Phase of Value Engineering*) [Garza &

Alcantara, 1997] consiste em dois módulos: KRM (*Knowledge Representation Module*) e RSM (*Rationale Storage Module*). O primeiro contém os objetos e os atributos que representam a informação de projeto e depende do domínio em que o projeto é executado. O segundo é independente de domínio, usado para capturar, armazenar e recuperar DR e, portanto, contém todas as decisões de projeto tomadas em relação a parâmetros de performance dos objetos do KRM. O DRIVE é um sistema baseado em regras e o DR é capturado de maneira estruturada. Esse sistema processa o DR capturado, agindo como um assistente de verificação de dados para o projetista. Além disso, tanto projetistas como sistemas de computador podem interpretar o DR capturado. Apesar de mencionar que o sistema é capaz de recuperar o DR capturado, não são dados maiores detalhes de como isso acontece.

O REMAP/MM [Ramesh & Sengupta, 1995] é o sistema mais parecido com a DocRationale. É um sistema hipermídia de suporte à decisão que facilita a captura de diferentes tipos de DR utilizando múltiplas mídias, assim como acontece na DocRationale. Oferece também suporte à captura de DR a partir da utilização de um modelo baseado no IBIS para a representação do DR. De maneira geral, a captura é realizada via uma interface gráfica que permite às equipes conduzirem suas deliberações, e um editor multimídia permite a criação e edição de documentos hipermídia. Uma diferença desse sistema para a DocRationale é o fato do REMAP/MM exigir a utilização de um editor específico para criar os documentos hipermídia, o que é feito de maneira livre da DocRationale. Os documentos são editados em quaisquer ferramentas e seus *uploads* feitos posteriormente na DocRationale. No REMAP/MM é utilizada uma linguagem de consulta para definir os vários tipos de consultas que podem ser realizadas para buscar as informações desejadas. Uma interface gráfica mostra as consultas e a informação retornada.

A principal vantagem do mecanismo de recuperação proposto para a DocRationale, em relação à recuperação no REMAP/MM, é que a consulta fica transparente ao usuário. O usuário tem apenas que fornecer as palavras-chave que serão buscadas nos documentos.

4.10 Considerações Finais

Diversos sistemas para automatizar a captura e a recuperação de DR são encontrados na literatura. Neste capítulo foi apresentada a DocRationale, uma ferramenta *Web* desenvolvida para permitir a captura, armazenamento e recuperação de DR. A CoTeia e o GroupNote foram integrados à DocRationale para apoiar a captura de DR, oferecendo funcionalidades como a

criação e edição de páginas *Web*, o *upload* de arquivos e as anotações. Os dados dos projetos, assim como os DRs relacionados aos artefatos de software, são armazenados em bases de dados relacionais distribuídas: os dados dos projetos são armazenados na base de dados da DocRationale, as páginas dos projetos e os arquivos anexos são armazenados na base de dados da CoTeia e na base de dados do GroupNote são armazenados os DRs dos artefatos.

Uma análise de uso da DocRationale apontou algumas deficiências como não ser facilmente compreendida e não ter suas funcionalidades descobertas de maneira simples. Problemas de interação também foram indicados. Visando minimizar essas deficiências, foram apresentadas algumas melhorias. A primeira delas foi elaborar documentos de ajuda para facilitar a utilização e o entendimento da ferramenta. A segunda foi acrescentar uma funcionalidade para inserir automaticamente os projetos a partir de documentos XML.

Outra melhoria sugerida para a DocRationale foi o aperfeiçoamento na representação de DR. Com essa nova funcionalidade, a ferramenta poderia oferecer diferentes opções para representar o DR capturado. Para isso, no entanto, foram elaborados documentos XML Schema para definir a estrutura da representação linear dos modelos mais comumente utilizados na argumentação: IBIS, PHI, QOC e DRL. A aquisição do DR seria realizada de maneira semelhante à inserção automática dos projetos, isto é, via documentos XML.

No entanto, a principal deficiência/limitação da DocRationale, relacionada a este trabalho, é oferecer apenas a navegação para a recuperação de DR. Conforme a quantidade de projetos cresce, essa estratégia torna a recuperação do DR bastante custosa para o usuário. Procurando viabilizar uma nova estratégia, além das estratégias de recuperação nos sistemas de DR, realizou-se um estudo sobre as tecnologias *Web* que poderiam oferecer suporte a essa nova estratégia para a ferramenta. Esse estudo é apresentado no próximo capítulo.

CAPÍTULO

5

Tecnologias de Suporte à Recuperação de DR

5.1 Considerações Iniciais

Com o crescimento da *Web* também cresceu o número de tecnologias e ferramentas desenvolvidas para aplicação nesse ambiente.

Dentre as diversas tecnologias e ferramentas disponíveis, algumas delas foram selecionadas para compor o mecanismo de recuperação de DR proposto para a DocRationale. Além de desempenharem as funções desejadas para o mecanismo de recuperação, essas ferramente tinham que, adicionalmente, ser integráveis à DocRationale. Dentre as tecnologias e ferramentas *Web* estudadas, pode-se citar: linguagem XML, *framework* Cocoon, eXist e algumas ferramentas de busca.

5.2 XML e XML Schema

XML (*Extensible Markup Language*) é uma linguagem de marcação projetada para descrever dados, que foi desenvolvida por um grupo dentro do *World Wide Web Consortium* (W3C) em

1996. São objetivos da linguagem [Bray et al., 1997]: a) ser utilizável na Internet, b) oferecer suporte a diversas aplicações, c) possuir o mínimo possível de características opcionais, d) ser clara e legível, e) ser formal e concisa, f) ser fácil de criar documentos utilizando a linguagem, g) permitir que documentos criados com XML sejam facilmente processados.

Resumidamente, a XML é utilizada para a representação digital de documentos. Isso significa que os documentos têm que ser legíveis para os computadores para que eles possam armazenar, processar, buscar, transmitir e mostrar esses documentos [Goldfarb, 1998].

Na Figura 5.1 é apresentado um exemplo simples de um documento XML.

```
<?xml version="1.0" encoding="ISO-8859-1?>
<mensagem id="5074">
  <de> Maria </de>
  <para> João </para>
  <conteúdo>
    Não esqueça da reunião hoje.
  </conteúdo>
</mensagem>
```

Figura 5.1: Documento XML simples

O documento começa com uma instrução de processamento: `<?xml . . . ?>`. Essa instrução é o mecanismo de inserção de informações explícitas em um documento, que são destinadas a alguma aplicação. No exemplo, as informações são sobre a versão e o conjunto de caracteres utilizados (nesse caso, caracteres para a maioria das línguas da Europa ocidental, inclusive o português).

Os elementos são delimitados por < > e identificam a natureza do conteúdo que fica entre a *start-tag* (`<elemento>`) e a *end-tag* (`</elemento>`). Os elementos vazios, ou seja, que não possuem conteúdo, são apresentados apenas em uma *end-tag*. Os nomes dos elementos são *case sensitive*, podendo conter letras, números e outros caracteres, exceto o espaço em branco. Além disso, os nomes dos elementos não devem iniciar com **XML** e todas as suas variações.

Os elementos XML podem ter atributos, que são pares nome-valor que ocorrem dentro da *tag* depois do nome do elemento. No exemplo `<mensagem id="5074">`, **mensagem** é o elemento com o atributo **id** cujo valor é **5074**.

Os comentários em um documento XML começam com `<!-` e terminam com `->`. Os comentários podem conter qualquer tipo de dado exceto a seqüência "–".

Um documento XML também pode conter seções `CDATA`. Elas são úteis quando se deseja

que todos os caracteres de um texto sejam interpretados como caracteres e não como elementos de marcação. Por exemplo, textos contendo os caracteres <, >, & etc., comuns em trechos de código de programas ou em textos descrevendo a XML.

A partir dessa descrição, percebe-se que as *tags* não são pré-definidas. Essa é uma das grandes vantagens da XML: *tags* próprias podem ser definidas.

A linguagem XML possui algumas regras de sintaxe que são apresentadas, resumidamente a seguir:

- Todos elementos XML devem ter uma *tag* de fechamento (<mensagem> ... </mensagem>).
- Elementos XML são *case sensitive* (<mensagem> é diferente de <Mensagem>).
- Todos elementos XML devem ser aninhados apropriadamente (<mensagem> <conteúdo> ... </conteúdo> </mensagem>).
- Todos documentos XML devem ter um elemento raiz.
- O espaço em branco é preservado.
- Valores de atributos devem estar entre aspas (<mensagem id="5074"> e não <mensagem id=5074>).

Essas regras definem se um documento XML é bem formado ou não. Assim, se um documento segue todas as regras de sintaxe da XML, esse documento é considerado bem formado.

Para estruturar os documentos XML, o *XML Schema Definition* (XSD ou XML Schema) é uma das opções que pode ser utilizada. Um XML Schema define: a) os elementos que podem aparecer em um documento, b) os atributos que podem aparecer em um documento, c) quais elementos são elementos filhos, d) a ordem dos elementos filhos, e) o número de elementos filhos, f) se um elemento é vazio ou pode incluir algum texto, g) os tipos de dados para elementos e atributos, h) valores *default* e fixos para elementos e atributos.

Além dessas definições, também é possível incluir restrições, tipos de elementos e grupos. As restrições são usadas para controlar valores aceitáveis para elementos XML ou atributos. A definição de tipos de elementos é utilizada quando se quer definir um elemento e não existem tipos pré-definidos que sejam adequados. Os grupos são usados para definir conjuntos de elementos relacionados.

Através do XML Schema é possível dizer se um documento XML é válido ou não. Um documento válido é um documento bem formado e que também está em conformidade com um XML Schema.

5.3 Cocoon

O Projeto Cocoon [Cocoon, 2004] foi fundado em 1999 como um projeto de código aberto sob o *Apache Software Foundation*. É um *framework* de desenvolvimento *Web* construído ao redor do conceito de desenvolvimento *Web* baseado em componentes. Esse conceito é implementado no Cocoon em torno da noção de *pipelines* de componentes.

Uma *pipeline* é uma cadeia de processamento que determina as ações do Cocoon quando um documento é requisitado. Uma *pipeline* pode ser simples ou mesmo bastante complexa. Toda *pipeline* inicia com um *Generator*, continua com ou sem *Transformers*, e termina com um *Serializer*. Cada componente na *pipeline* se especializa em uma operação particular.

O *sitemap* é parte mais importante do Cocoon. Ele contém as declarações dos componentes e as funções configuradas que uma aplicação baseada no Cocoon provê. O arquivo *sitemap* é um documento XML que consiste em duas principais áreas: uma biblioteca de componentes e uma área de definição de documento. A área de definição de documento utiliza os componentes configurados para descrever como um documento deve ser gerado. Essas duas áreas lógicas são divididas em diversas seções. A estrutura global do *sitemap* é formada por cinco seções: *components*, *views*, *resources*, *action-sets* e *pipelines* [Langham & Ziegeler, 2001].

A seção de componentes é a mais importante. Ela contém os componentes disponíveis. Como mencionado anteriormente, a maneira mais comum de construir *pipelines* é utilizar um *Generator*, um ou mais *Transformers* e um *Serializer*. A Figura 5.2 mostra como é utilizado o conceito de *pipeline* no Cocoon.

O *Generator* obtém os dados de uma fonte de dados, converte-os se necessário em um documento XML, e o torna disponível para continuar o processamento na *pipeline*. O dado que é acessado pelo *Generator* não precisa estar no formato XML. No entanto, quando ele termina sua tarefa, o formato que é trocado entre os componentes da *pipeline* é XML.

O *Transformer* é um componente opcional que pode ser usado na *pipeline* depois do *Generator*. O *Transformer* recebe o documento XML de um *Generator* ou de outro *Transformer*,

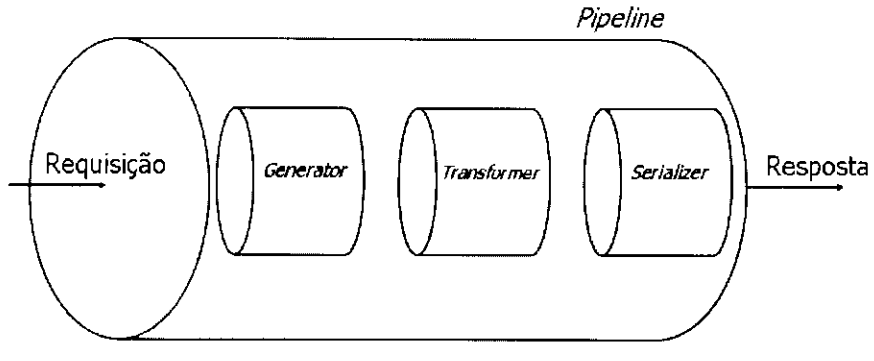

Figura 5.2: ***Pipeline*** **no Cocoon [Cocoon, 2004]**

caso ele não seja o primeiro na *pipeline*. Um *Transformer* não tem que transformar todo o documento. Ele pode focalizar apenas os elementos XML que ele precisa para executar sua tarefa. Assim, se os dados XML contém *tags* que são específicas para o *Transformer* atual, ele avalia essa informação e age sobre ela para obter o conteúdo adicional ou manipular o documento XML de alguma outra maneira. Assim que o *Transformer* executa sua tarefa, ele passa o documento XML para o próximo componente. Isso significa que o documento originalmente lido pelo *Generator* pode conter *tags* específicas para vários *Transformers*. Também é possível um componente gerar comandos dinamicamente para componentes à sua frente na *pipeline*.

Toda *pipeline* deve terminar com um *Serializer*. O *Serializer* recebe o documento XML gerado e transformado e o coloca no formato requerido. O *Serializer* também pode adicionar informação específica ao documento de modo que a aplicação de destino possa mostrá-lo corretamente.

Resumidamente: a partir de uma requisição, dados são processados na *pipeline* e o resultado final é um documento no formato escolhido.

5.4 eXist

eXist [Meier, 2002] é uma base de dados XML de código aberto, independente de plataforma, completamente escrita em Java, que provê o armazenamento de documentos XML em coleções hierárquicas, de maneira semelhante a arquivos armazenados em diretórios.

Como características desse banco de dados XML, pode-se citar:

- O armazenamento de dados é baseado em árvores B+ e em arquivos paginados. Os nós

dos documentos são armazenados em um DOM [1] persistente.

- Os documentos são gerenciados em coleções hierárquicas. As coleções não têm que estar de acordo com um esquema ou tipo de documento pré-definido.
- A indexação dos documentos é baseada em um esquema numérico que oferece suporte à identificação rápida de relacionamentos estruturais entre os nós, tais como pai/filho e ancestral/descendente.
- A criação automática de índices é feita por *default*. São utilizados índices estruturais para os nós elementos e atributos, e um índice completo para textos e valores dos atributos. A indexação completa pode estar ligada ou desligada para partes específicas de um documento. A manutenção dos índices estruturais é realizada automaticamente.
- O eXist tem sua própria máquina de consulta: XQuery. Diferentemente das implementações convencionais, a máquina de consulta do eXist tenta evitar caminhos *top-down* ou *bottom-up* nas árvores. Ao invés disso, ela utiliza algoritmos de caminhos para computar os relacionamentos dos nós.
- As atualizações são em nível de documentos e nós.
- O **XML-RPC** (*XML Remote Procedure Call*) é o protocolo preferido e usado pelos *drivers* XML:DB API. Tal protocolo provê acesso total a todas as funções da base de dados.

Devido às suas características, o eXist pode ser utilizado em aplicações com grandes ou pequenas coleções de dados. A página *Web* do eXist é mostrada na Figura 5.3.

Para a busca de documentos, foram implementadas extensões na XPath, que é uma linguagem de consultas XML. Essa linguagem utiliza caminhos (*path*) para navegar através da estrutura hierárquica dos documentos XML, mas possui limitações para buscar uma cadeia de caracteres (*string*) dentro do conteúdo referente a um elemento. Isso faz com que a linguagem não seja adequada se a busca for realizada em documentos que contêm grandes seções de texto.

Considerando esse problema, o eXist oferece dois operadores e várias funções adicionais para possibilitar acesso a todo conteúdo dos documentos. O operador `&=` seleciona os nós (elementos) que contêm todos os termos passados como argumento em uma consulta. O operador

[1] DOM (*Document Object Model*) é uma interface que permite programas e *scripts* acessarem e atualizaram o conteúdo, a estrutura e o estilo dos documentos. O documento pode ser processado e os resultados desse processamento pode ser incorporado na página a ser apresentada.

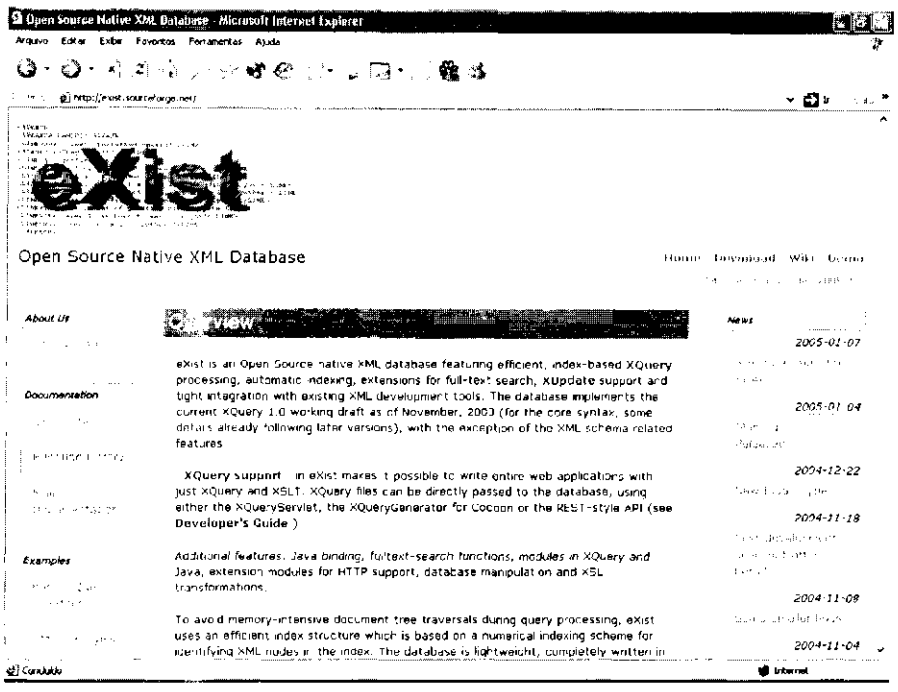

Figura 5.3: Página *Web* do eXist

|=, por sua vez, encontra os nós que possuem qualquer um dos termos passados no argumento. A ordem dos termos não é importante, mas é possível impor uma ordem dos termos a serem buscados. Também é possível procurar por termos, definindo a proximidade entre eles. Por exemplo, pode-se buscar as palavras "documento" e "projeto" que estejam separadas por outras dez palavras. Além disso, com as extensões efetuadas para oferecer suporte à busca de documentos, podem ser realizadas consultas com qualquer combinação de coleções e documentos.

5.5 Ferramentas de Busca

Algumas ferramentas de busca foram estudadas como possíveis ferramentas a serem utilizadas para a busca em documentos HTML e mesmo em documentos XML. Como principais características, elas deveriam: (i) ter código aberto ou ser livre, (ii) oferecer versão para o sistema operacional Linux, (iii) buscar por palavras ou frase, e (iv) ser integrável à DocRationale.

Dentre as diversas ferramentas de busca existentes, todas as descritas a seguir, **mnoGoSearch**, **ASPSeek**, **Swish-e** e **DataparkSearch Engine**, possuem essas características.

5.5.1 mnoGoSearch

mnoGoSearch [mnoGoSearch.org, 2004] é uma máquina de busca *Web* completamente baseada em SQL (*Structural Query Language*). Consiste em duas partes, que são o mecanismo de indexação e o *front-end*. A primeira percorre as referências de hiperdocumentos HTML e armazena as palavras encontradas e novas referências na base de dados. A segunda é um *front-end* CGI (*Common Gateway Interface*) que proporciona a busca dos dados coletados pelo indexador.

Dentre as diversas caracterísicas da mnoGoSearch estão:

- Oferecer suporte para os protocolos HTTP, HTTPS e FTP.
- Possuir *parsers* externos para oferecer suporte a outros tipos de documentos além dos documentos HTML.
- Buscar todas as formas de palavras (variações da mesma palavra, palavras sinônimas).
- Buscar frases.
- Oferecer suporte à linguagem de consulta booleana.
- Ordenar os resultados por relevância e popularidade.
- Poder ser utilizada com diversas bases de dados: mySQL, PostgreSQL, SQLite, Interbase, Oracle.

Dois recursos importantes da mnoGoSearch, considerando a DocRationale, são o **PHP Extension Module** e o **PHP *Front-End***. O primeiro é um módulo que permite o acesso à base de dados da mnoGoSearch a partir de um programa PHP (que é o caso da DocRationale), e então torna possível escrever aplicações de busca utilizando a linguagem PHP. O segundo é uma aplicação de busca, escrita em PHP, que utiliza o PHP Extension Module.

5.5.2 ASPSeek

ASPSeek [aspseek.org, 2004] é uma máquina de busca de Internet. Consiste em um programa de indexação, um *daemon* de busca e um *front-end* CGI.

O programa de indexação percorre os *sites* e armazena as página encontradas em estruturas de dados especiais. O *daemon* aguarda as buscas e as executa assim que são passadas pelo *front-end*, que também formata os resultados em uma página HTML.

Algumas características da ASPSeek são:

- Possuir a habilidade de indexar e procurar milhares de documentos.
- Oferecer suporte aos protocolos HTTP e HTTPS.
- Ter capacidade de procurar por palavras e frases.
- Prover buscas em documentos HTML assim como em documentos texto simples.
- Ordenar os resultados por relevância.
- Possibilitar a customização dos resultados das buscas.

Comparativamente, a ASPSeek oferece menos recursos que a mnoGoSearch.

5.5.3 Swish-e

Swish-e [swishe.org, 2004] é uma máquina de busca que pode rápida e facilmente indexar diretórios de arquivos ou *websites* e buscar os índices gerados. Também pode indexar páginas *Web*, arquivos texto ou dados armazenados em uma base de dados relacional.

Suas principais características são:

- Indexar documentos de diferentes formatos, incluindo HTML e XML.
- Indexar outros tipos de arquivos utilizando filtros.
- Oferecer suporte ao protocolo HTTP.
- Buscar por palavras e frases.
- Ordenar por relevância os resultados da busca.
- Poder ser customizada.

Para utilizar a Swish-e é preciso configurar a Swish-e para criar os índices para os documentos, criar um índice executando o Swish-e, e iniciar uma interface como um *script* CGI para buscar o índice e mostrar os resultados. Dentre as máquinas de busca abordadas nesta seção, a Swish-e é a mais simples.

5.5.4 DataparkSearch Engine

DataparkSearch Engine [dpSearch, 2004] é uma máquina de busca *Web* projetada para realizar buscas em *websites*, grupos de *websites*, intranet e sistema local. Consiste em duas partes: o mecanismo de indexação que percorre as referências dos hiperdocumentos e armazena as palavras encontradas e novas referências na base de dados, e um *front-end* CGI para prover a busca utilizando os dados coletados pelo mecanismo de indexação.

Algumas características dessa máquina de busca são:

- Oferecer suporte para os protocolos HTTP, HTTPS e FTP.
- Oferecer suporte *built-in* para HTML, XML, texto, mpeg.
- Buscar por palavras e frases.
- Ordenar os resultados da busca por relevância, popularidade, última data de modificação e importância (valor da relevância multiplicada pela popularidade).
- Oferecer suporte aos banco de dados: MySQL, PostgreSQL, unixODBC, Interbase, Oracle (8 e 8i).

De maneira semelhante à mnoGoSearch, a DataparkSearch Engine oferece os recursos PHP Extension Module e PHP *Front-End*.

A partir das funcionalidades oferecidas pelas tecnologias e ferramentas descritas, foi possível definir o mecanismo de recuperação de DR para a DocRationale.

5.6 Considerações Finais

O advento da *Web* propiciou o aparecimento de diferentes tecnologias assim como o desenvolvimento de diversas ferramentas *Web*, especialmente com a sua expansão.

Dentre essas tecnologias e ferramentas, algumas foram consideradas para serem adotadas no mecanismo proposto neste trabalho para a recuperação de DR. Além de oferecerem funcionalidades desejáveis ao mecanismo de recuperação, essas ferramentas eram integráveis à DocRationale.

A linguagem XML foi uma das tecnologias analisadas. Documentos XML são legíveis para os computadores, podendo ser armazenados, processados, buscados e exibidos. Já um documento XML Schema é utilizado para definir a estrutura de um documento XML: quais elementos e atributos podem aparecer no documento, quais serão os seus tipos e valores iniciais, quais são elementos filhos etc.

Outra tecnologia de suporte à recuperação de DR estudada foi o Cocoon, um *framework* de desenvolvimento *Web* construído ao redor do conceito de desenvolvimento *Web* baseado em componentes, implementado em torno da nação de *pipelines*.

Para armazenar e buscar documentos XML, uma opção analisada neste trabalho foi o eXist, uma base de dados XML de código aberto, independente de plataforma, que provê o armazenamento de documentos XML em coleções hierárquicas, de maneira semelhante a arquivos armazenados em diretórios. Além de gerenciar os documentos XML, o eXist também oferece poderosos recursos para a busca nesses documentos.

Por fim, quatro ferramentas de busca em documentos HTML foram consideradas. Todas essas ferramentas tinham código aberto ou eram livre, ofereciam versão para o sistema operacional Linux, realizavam buscas por palavras ou frases, e eram integráveis à DocRationale. As características particulares de cada uma delas também foram apresentadas.

Resumidamente, foram apresentadas neste capítulo, os benefícios e as funcionalidades oferecidas por tecnologias que vêm de encontro com as funções exigidas para a recuperação de DR na DocRationale. No próximo capítulo é mostrado como eles podem ser utilizados para melhorar a recuperação na ferramenta DocRationale.

CAPÍTULO

6

Mecanismo para Recuperação de DR

6.1 Considerações Iniciais

Apenas capturar e armazenar as informações não é o suficiente para que essas informações possam ser reutilizadas. Nesse sentido, a recuperação é de grande importância nos sistemas de DR, uma vez que ela possibilita a retomada da informação. Na DocRationale, a recuperação é bastante restrita, sendo possível somente navegar pelas informações. Essa característica da ferramenta pode tornar a busca por informações uma atividade bastante custosa, pois o usuário é obrigado a percorrer todas as informações disponíveis na ferramenta para tentar encontrar o que deseja.

Visando melhorar a busca de DR e a interação dos usuários com a DocRationale, investigou-se um mecanismo para ser adotado como estratégia de recuperação de DR na ferramenta [Fumagalli & Fortes, 2003]. A partir de estudos realizados sobre abordagens de recuperação de DR e de outras tecnologias que poderiam ser utilizadas nesse contexto, foi possível definir opções para esse mecanismo de recuperação de DR para a DocRationale. Tais opções são apresentadas neste capítulo.

6.2 Objetivo

O principal objetivo é investigar um mecanismo de busca de informações na DocRationale de maneira que a recuperação de DR seja mais eficiente do que a atualmente disponível na ferramenta, considerando também opções de busca que o usuário queira considerar, como selecionar um projeto específico no qual a recuperação de DR será realizada [Fumagalli & Fortes, 2004].

Dentre as abordagens de recuperação de DR estudadas, a abordagem escolhida para a ferramenta DocRationale foi a recuperação baseada em consulta através de busca por palavra-chave.

Para isso, a idéia é integrar os dados armazenados nas diferentes bases de dados (DocRationale, GroupNote e Coteia) de maneira a agregá-los em um único documento para cada projeto. A linguagem de marcação XML foi escolhida para criar os documentos. Uma vez criados, os documentos XML de cada projeto, palavras e frases devem ser buscadas nesses documentos com o propósito de promover a recuperação dos DRs inicialmente armazenados na DocRationale. Algumas tecnologias e ferramentas foram utilizadas para compor diferentes soluções que pudessem definir esse mecanismo. Deve-se ressaltar, entretanto, que nenhuma dessas opções foi efetivamente implementada na ferramenta.

6.3 Definição da Estrutura dos Documentos XML

Devido às diversas vantagens oferecidas pela XML, como definição de *tags* específicas e a presença de significado embutido nas mesmas, essa linguagem foi escolhida para criar os documentos de integração dos dados e do DR da armazenados de maneira distribuída na DocRationale. No entanto, antes de serem efetivamente criados, foi necessário definir a estrutura que os documentos XML deveriam possuir. Para isso foi elaborado um documento XML Schema [W3C, 2001b]. As páginas *Web* dos projetos, disponíveis na DocRationale, foram utilizadas para definir o XML Schema, de maneira semelhante ao processo descrito por Fumagalli et al. [2003] para abstrair a estrutura de documentos.

Nesse processo, o primeiro passo é abstrair uma estrutura candidata. Para isso é necessário: (i) Ler as páginas *Web* dos projetos, (ii) Identificar os elementos candidatos, (iii) Definir os elementos candidatos, (iv) Realizar uma análise heurística, (v) Confrontar a estrutura com a estrutura das páginas *Web*, (vi) Definir uma estrutura candidata. O passo seguinte é rever essa estrutura quantas vezes forem necessárias até chegar a uma estrutura definitiva. A partir desse

ponto, a tarefa é definir o documento que represente essa estrutura. Os documentos devem ser reescritos seguindo a estrutura intermediária definida, e serem revisados até que o documento de definição seja definitivo.

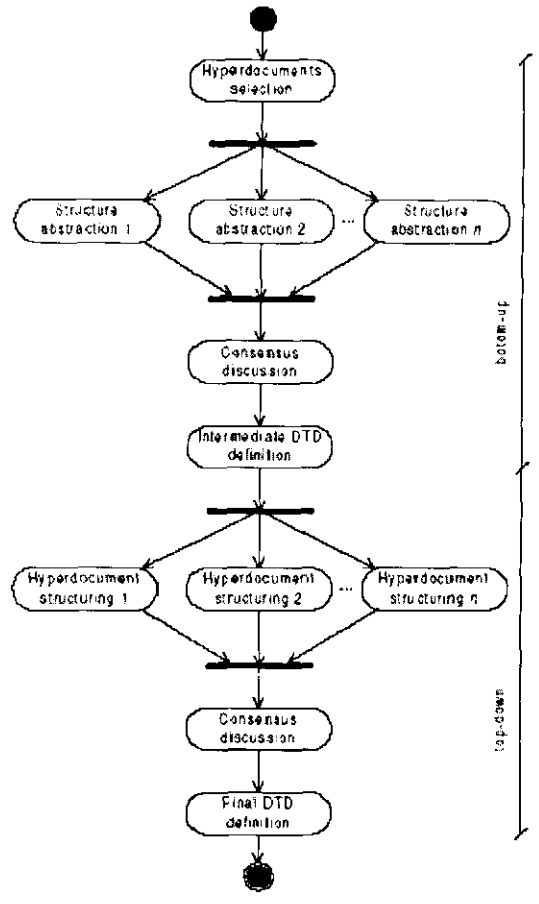

Figura 6.1: Processo para abstrair a estrutura dos documentos XML [Fumagalli et al., 2003]

Na Figura 6.1 que mostra o processo de abstração de estrutura de documentos, a tarefa **Consensus discussion** foi substituída pela revisão das estruturas pois o processo foi executado por apenas uma pessoa, impossibilitando a discussão.

Seguindo esse processo, obteve-se como resultado o XML Schema (vide Apêndice C) que define a estrutura dos documentos XML dos projetos. Os documentos XML devem estar em conformidade com esse documento XML Schema.

6.4 Criação dos Documentos XML

Como já mencionado, para cada projeto criado na DocRationale, deve existir um documento XML correspondente. Os elementos definidos no XML Schema devem ser inscridos nos documentos e os dados, provenientes das consultas realizadas nas bases de dados e relacionados a cada um dos elementos, devem ser também inseridos no documento XML. Deve-se ressaltar que a estrutura definida no XML Schema deve ser mantida, pois ela reflete a hierarquia das informações referentes aos DRs obtidos. Esses documentos devem ser recriados de tempos em tempos para que quando eles forem consultados, não estejam desatualizados.

Para a criação desses documentos foram consideradas duas opções: utilizar o *framework* Cocoon ou implementar um programa em uma linguagem que possa acessar o MySQL.

6.4.1 Criação dos Documentos XML utilizando Cocoon

Os documentos XML são gerados pelo Cocoon através do processamento de um documento XSP. XSP (*Extensible Server Pages*) [XSP, 2004] é uma linguagem de marcação dinâmica como JSP (*Java Server Pages*) ou ASP (*Active Server Pages*) que provê diretivas de código embutido, e é fortemente integrada com o Cocoon. Um documento XSP é um documento XML com *tags* especiais XSP, que pode ser utilizado como o ponto de partida de uma *pipeline* de processamento do Cocoon. O documento XSP é processado por um *Generator* especial do Cocoon chamado *ServerPagesGenerator*, cujo processador converte o documento XSP em um programa fonte para uma particular linguagem de processamento. Através da execução do programa gerado, o documento XML resultante, descrito pelo documento XSP original, é gerado e intercalado com o conteúdo dinâmico gerado pelo código extraído a partir das diretivas de *tags* XSP. Em outras palavras, o documento XSP é utilizado pelo *ServerPagesGenerator* para criar um documento XML. Na Figura 6.2 é apresentado um exemplo de um documento XSP.

Nesse exemplo é mostrado como estabelecer a conexão com a base de dados MySQL da ferramenta DocRationale, executar a consulta SQL definida, retornando os resultados obtidos. A *logicsheet* **esql** é uma *logicsheet* XSP que executa as consultas SQL e serializa os resultados em um documento XML.

Entretanto, antes que o processamento do documento XSP seja realizado, é necessário configurar alguns documentos do Cocoon. Inicialmente, é necessário copiar o *driver* do MySQL

```
<xsp:page language="java"
xmlns:xsp="http://apache.org/xsp"
xmlns:esql="http://apache.org/cocoon/SQL/v2">
  <DocRationale>
    <esql:connection>
      <esql:pool>doc_rat</esql:pool>
       <esql:execute-query>
         <esql:query>SELECT * FROM project</esql:query>
           <esql:results>
             <esql:row-results>
               <project>
                 <esql:get-columns/>
               </project>
             </esql:row-results>
           </esql:results>
       </esql:execute-query>
     </esql:connection>
  </DocRationale>
</xsp:page>
```

Figura 6.2: Documento XSP

no diretório de bibliotecas do Cocoon, caso ele não exista. A seguir, deve-se referenciar esse *driver* no arquivo **web.xml**. O passo seguinte é definir a conexão com o banco de dados MySQL utilizado na DocRationale no arquivo **cocoon.xconf**. Um exemplo dessa definição é mostrado na Figura 6.3.

```
<datasources>
  <jdbc name=doc_rat>
    <pool-controller min="5" max="10"/>
      <dburl>
         jdbc:mysql://143.107.183.160/db_docrat?autoReconnect=true
      </dburl>
      <user> user </user>
      <password> password </password>
  </jdbc>
</datasources>
```

Figura 6.3: Definição da conexão do Coccon com o banco de dados MySQL

Nesse exemplo, a conexão com o banco de dados é identificada por **doc_rat**. São definidos a URL onde se encontra base de dados e o seu respectivo nome. Além disso, também é estabelecido o acesso à base, informando o usuário e a senha. Assim, no documento XSP é necessário apenas que o nome que identifica a conexão com banco de dados seja informado.

Como último passo, deve-se configurar o *sitemap*, definindo os componentes que serão utilizados e o processamento a ser executado na *pipeline*.

Resumidamente, ao criar os documentos XML utilizando o Cocoon, o documento XSP é

o responsável por acessar as bases de dados da DocRationale, CoTeia e GroupNote e realizar as consultas necessárias para recuperar os dados corretos, seguindo a estrutura definida para o documento XML. O Cocoon processa esse documento XSP e gera os documentos XML correspondentes aos projetos existentes na ferramenta.

6.4.2 Criação dos Documentos XML utilizando Programas

Os documentos XML podem ser gerados a partir da execução de um programa implementado em qualquer linguagem de programação que tenha métodos de acesso ao MySQL, por exemplo, C, C++, Python, Lisp, Smalltalk, Java, PHP.

O programa deve acessar as bases de dados da DocRationale, CoTeia e GroupNote, realizar as consultas, utilizando os elementos e a estrutura definidos no XML Schema, e os resultados das consultas obtidos para gerar os documentos XML de cada um dos projetos. Esses documentos devem ser gerados de tempos em tempos de modo a mantê-los atualizados.

Uma vez criados, os documentos XML dos projetos têm que ser armazenados para que possam ser utilizados na recuperação do DR capturado e armazenado na DocRationale (transcrito agora nos documentos XML).

6.5 Armazenamento e Busca dos Documentos XML

Para o armazenamento dos documentos XML dos projetos capturados na ferramenta DocRationale, foram consideradas duas alternativas: sistemas de arquivos e banco de dados eXist. Em relação à primeira alternativa, é preciso apenas criar um diretório no qual os documentos XML ficarão armazenados e definir o caminho desse diretório para que as buscas sejam realizadas nesses documentos. No segundo caso, é preciso que inicialmente, os documentos gerados sejam armazenados no eXist. As consultas são realizadas diretamente no banco de dados.

As descrições mais detalhadas de como isso pode ser feito são apresentadas na seção seguinte, que mostra algumas opções consideradas para a definição do mecanismo de recuperação de DR da DocRationale.

6.6 O Mecanismo

Como mencionado anteriormente, a estratégia escolhida para o mecanismo de recuperação de DR na DocRationale foi a baseada em consulta, através da busca por palavra-chave.

Além disso, deve-se lembrar que a integração da DocRationale com a Coteia possibilitou a criação automática de páginas *Web* de projetos e de artefatos relacionados aos projetos. A partir das páginas *Web* dos artefatos, arquivos complementares a esses artefatos podem ser anexados. Assim, o mecanismo definido deve possibilitar a busca também nesses arquivos.

Portanto, considerando a estratégia escolhida, algumas opções para esse mecanismo foram definidas. Tais opções são descritas a seguir.

6.6.1 Opção 1: Cocoon e eXist

Nessa opção, utiliza-se o Cocoon para gerar os documentos XML e o eXist para o armazenamento e a busca desses documentos. O eXist foi escolhido porque além de armazenar documentos XML, ele também possibilita a combinação de diversos tipos de documentos (o que é importante quando os usuários têm a opção de escolher em quais documentos realizar a busca) e a definição de diversos tipos de consultas, o que pode facilitar a busca dos DRs transcritos no documento XML de cada projeto. Em relação ao Cocoon, ele foi escolhido porque além de ser parte integrante do eXist, é possível via Cocoon acessar as bases de dados MySQL e gerar documentos XML.

Geração dos Documentos

Inicialmente, é preciso implementar um programa *daemon* para gerar os documentos XML de tempos em tempos. Como o Cocoon foi desenvolvido em Java, pode-se optar por essa linguagem. Assim, esse *daemon* é o responsável por disparar as requisições no Cocoon para que os documentos XML sejam gerados.

Também é preciso converter os arquivos anexos assim que seu *upload* é realizado na DocRationale. Para isso, pode-se usar, por exemplo, as ferramentas descritas a seguir.

- wvWare [Lachowics, 2000] é uma aplicação, com uma série de opções de linhas de comando, provida com a biblioteca **wv** que permite acesso a arquivos Microsoft Word, com

o propósito de convertê-los em outros formatos. Dentre os diversos formatos para os quais os arquivos Word podem ser convertidos estão: HTML 4.0, Latex, dvi, PS, PDF, TXT, WML, RTF.

- pdftohtml [Kruk, 2003] é uma ferramenta que converte arquivos do formato Portable Document Format (PDF) nos formatos HTML e XML. É bascada no XPDF, um visualizador de código aberto para arquivos pdf.

- txt2html [Andersen, 2001] é uma ferramenta que converte arquivos em formato de texto para arquivos HTML. Ela oferece suporte para cabeçalhos, listas, tabelas, caracteres de marcação simples e hyperlinks.

- rtf-converter [Ford, 2003] é uma aplicação em linha de comando para converter arquivos de formato Rich Text Format (RTF) em HTML.

Na Figura 6.4 é representada a parte do mecanismo de recuperação de DR descrito anteriormente.

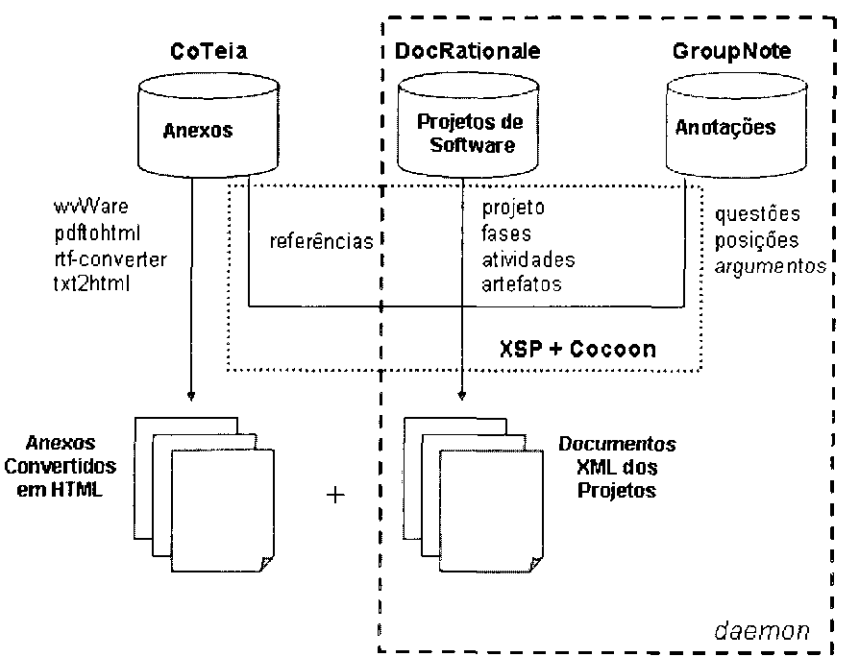

Figura 6.4: Geração dos documentos a serem utilizados na recuperação de DR

Gerados os documentos que serão utilizados na recuperação do DR, é preciso abordar como será seu armazenamento e busca.

Armazenamento e Busca dos Documentos

Pode-se utilizar novamente a linguagem Java, a partir de uma biblioteca XML-RPC (*XML Remote Procedure Call*), para inserir os documentos XML gerados no banco de dados eXist. Por exemplo, o método `parse` pode ser utilizado para essa inserção:

```
boolean parse(byte[] xml, String docName, int overwrite)
```

onde **xml** é a definição do conjunto de caracteres do conteúdo do arquivo (UTF-8, por exemplo), **docName** é o caminho do local onde o documento será armazenado na base de dados, e **overwrite** é definido como maior que zero para que a substituição de um arquivo existente no mesmo local seja realizada automaticamente.

Já os arquivos convertidos para HTML devem ser armazenados em um diretório no servidor *Web*. Uma vez armazenados, pode-se então realizar as buscas nesses documentos para recuperar o DR.

A busca nos documentos XML dos projetos é realizada através de consultas no próprio eXist. Como a ferramenta está implementada em PHP, a busca pelos DRs é realizada via uma classe PHP (PHP API) que possibilita acessar o eXist e realizar todo o tipo de consulta oferecida por ele.

O eXist pode conter um conjunto ilimitado de coleções [exist db.org, 2000]. Se uma API estiver sendo utilizada para consultar a base de dados, somente os documentos da coleção atual são processados por *default*. No entanto, existem quatro funções que podem mudar esse comportamento: **doc()**, **document()**, **collection()** e **xcollection**.

As funções **doc()** e **collection()** são funções padrão da XQuery. Já as funções **xcollection()** e **document()** representam extensões específicas do eXist. Essas funções são utilizadas para especificar o conjunto de documentos de entrada utilizados na consulta.

O eXist interpreta os argumentos para **collection()** e **doc()** como caminhos relativos ou absolutos, referenciando alguma coleção ou documento dentro da base de dados. Enquanto **doc()** é restrito a apenas um documento, **document()** aceita múltiplos caminhos de documentos que serão incluídos no conjunto de entrada. Se nenhum argumento for passado, todos os documentos são considerados.

A função **collection()** especifica a coleção cujos documentos serão utilizados na consulta. Por *default*, os documentos encontrados em sub-coleções da coleção especificada também são

incluídos. Isso significa que para duas coleções **/db/docrationale** e **/db/docrationale/testes**, **collection('db/docrationale')** incluirá todos os documentos encontrados em /db/docrationale e /db/docrationale/testes. Se for utilizado **xcollection('/db/docrationale')**, apenas os documentos encontrados em /db/docrationale serão incluídos.

O eXist oferece também funções e operadores adicionais para o acesso eficiente a todo conteúdo de um documento [exist db.org, 2000]. Por exemplo, a consulta:

```
//chapter[near(., 'XML database?', 50)]
```

irá retornar todos os capítulos contendo ambas as palavras na ordem correta e com menos de 50 palavras entre elas. Adicionalmente, o ? fará com que palavras no singular como no plural sejam procuradas. Além disso, a pesquisa é *case-insensitive*.

Existem ainda os operadores &= e |= que foram abordados no Capítulo 5.

Tendo visto todos os recursos oferecidos pelo eXist, é possível entender como é realizada um consulta no eXist. Um exemplo é apresentado a seguir.

```
document('/db/docrationale/doc1.xml','/db/docrationale/doc2.xml' ) //
      Artefact[art_name &="requisitos" AND art_name &= "modelo"]
```

Essa pesquisa seleciona nos documentos doc1.xml e doc2.xml todos os nós Artefato que possuem filhos art_name contendo um ou ambos os termos: "requisitos" e "modelo".

Sabendo como funciona a pesquisa no eXist, é possível através da PHP API definir consultas que considerem, por exemplo, apenas alguns projetos ou que procure por projetos cujo gerente é uma determinada pessoa, e assim por diante.

O resultado da busca, retornado como trechos de documentos XML, deve ser manipulado de maneira que esses trechos sejam exibidos na DocRationale como uma lista de *links* para os DRs recuperados.

Em relação aos documentos convertidos para HTML, pode-se utilizar máquinas de busca como a mnoGoSearch e a DataparkSearch Engine, que além de realizarem a busca nos documentos HTML, possuem um *front-end* PHP que pode facilitar a exibição dos resultados da busca na DocRationale.

Todo o mecanismo proposto na Opção 1 é apresentado na Figura 6.5

Além dessa, outras opções para o mecanismo de recuperação de DR na DocRationale foram consideradas.

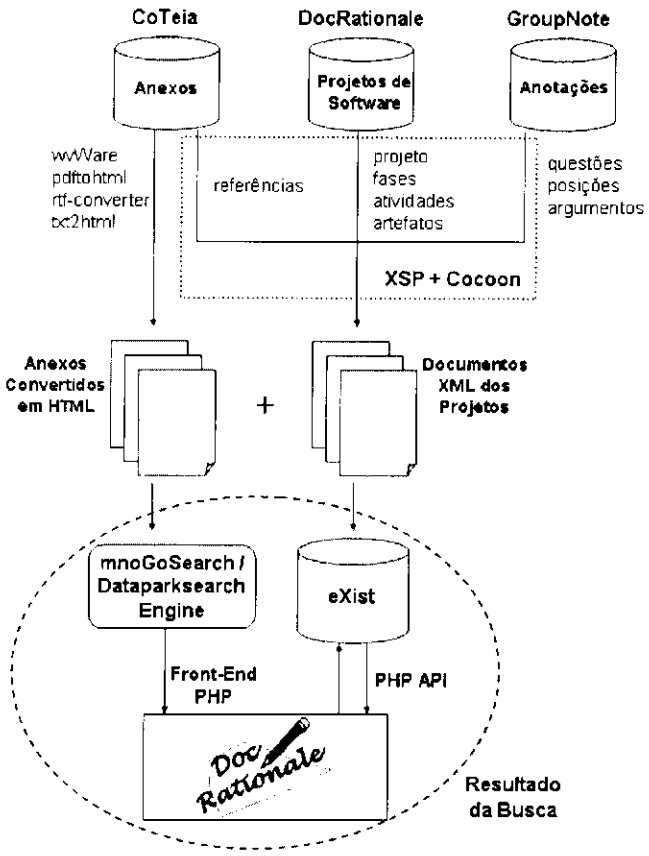

Figura 6.5: Opção para o mecanismo de recuperação de DR utilizando Cocoon e eXist.

6.6.2 Opção 2: Programa e eXist

Nessa opção, apenas o que muda em relação à proposta da Opção 1 é a utilização de um programa implementado em alguma linguagem de programação ao invés do uso do *framework* Cocoon.

A linguagem Java é uma boa opção nesse caso porque a DocRationale já tem como requisito o uso de uma máquina virtual Java e além disso, é considerada a melhor linguagem para se trabalhar com documentos XML. A descrição de um programa escrito em Java que poderia ser utilizado para gerar os documentos XML é feita a seguir.

Geração dos Documentos

Inicialmente, o programa acessa o banco relacional MySQL utilizando a API Java para JDBC, efetua as consultas e obtém os resultados. Para a criação efetiva dos documentos XML

é utilizada uma API DOM do Java: JDOM, que é uma implementação da especificação DOM para a linguagem Java. Resumidamente, essas duas APIs são as responsáveis por gerar todos os documentos XML dos projetos. Uma funcionalidade oferecida por esse programa é validar os documentos XML gerados a partir do XML Schema definido. Para isso, no entanto, existem módulos pré-definidos.

A conversão dos arquivos anexos para HTML continua sendo realizada como descrito na opção anterior.

Armazenamento e Busca dos Documentos

O armazenamento e a busca, tanto nos documentos XML quanto nos arquivos convertidos, são efetuados de maneira idêntica ao mencionado para a proposta do mecanismo de recuperação que utiliza o Cocoon e o eXist.

Na Figura 6.6 a diferença entre as Opções 1 e 2 é destacada.

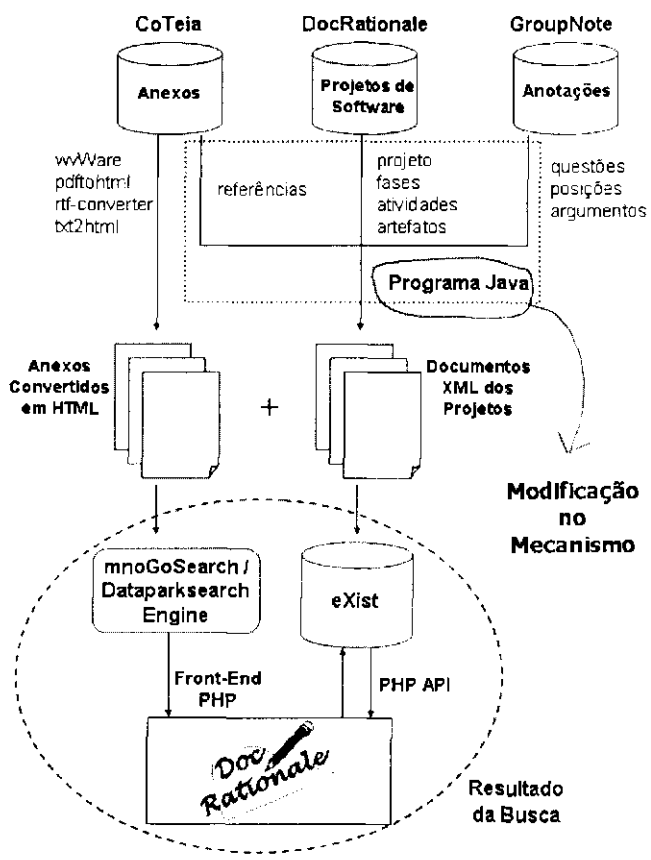

Figura 6.6: Opção para o mecanismo de recuperação de DR utilizando programa e eXist.

Além dessa opção, uma outra opção ainda pode ser definida. Na seção seguinte é descrita a última das opções consideradas.

6.6.3 Opção 3: Programa e Sistema de Arquivo

Nessa opção, apenas o que se mantém em relação à proposta da Opção 2 é a utilização de um programa implementado em alguma linguagem de programação. Para o armazenamento e a recuperação dos documentos XML, uma nova alternativa foi considerada.

Geração dos Documentos

Para a gerar os documentos XML é utilizado o mesmo programa definido para a Opção 2. A principal diferença está no fato de que os arquivos anexos não serão mais convertidos para HTML assim que forem acrescentados na DocRationale. Na sua busca, esses documentos serão diretamente convertidos e pesquisados por uma mesma ferramenta.

Armazenamento e Busca dos Documentos

Os documentos XML gerados serão mantidos em um sistema de arquivos. Os arquivos anexos são mantidos no diretório em que foram armazenados na DocRationale.

Para a realização da busca nesses documentos, foi escolhida para essa opção a máquina de busca Swish-e, uma vez que ela permite indexar arquivos de diretórios rápida e facilmente.

É possível configurar a Swish-e de maneira que as buscas possam ser realizadas nos documentos fontes declarados no arquivo de configuração. Também no arquivo de configuração da Swish-e podem ser definidas diretivas que controlam quais documentos devem ser indexados e como eles serão acessados. Por exemplo, o comando seguinte define que o método de acesso é a partir do sistema de arquivo:

```
swish-e swish.config -S fs
```

e a diretiva **IndexDir** define a fonte de documentos que usados na busca:

```
IndexDir ./index.html ./page1.html ./page2.html
```

Além dessas configurações, também é preciso configurar o indexador em relação ao conteúdo dos documentos. Assim sendo, a diretiva **IndexContents** pode escolher qual *parser*

utilizar, dependendo da extensão que o arquivo possui. Os *parsers* atualmente disponíveis na Swish-e percorrem documentos HTML, XML e TXT.

O exemplo a seguir define que o *parser* HTML deve ser utilizado quando forem encontrados documentos com as extensões .htm, .html e .shtml:

```
IndexContents HTML* .htm .html .shtml
```

A Swish-e também permite definir opções de filtros que convertem documentos de diversos formatos como PDF, PS e Microsoft Word em formatos para os quais existem *parsers* definidos.

Uma busca na Swish-e é definida basicamente da seguinte maneira:

```
swish-e -w foo AND bar
```

O comando -w indica que é uma consulta. Essa consulta realiza a busca em todos os documentos e retorna os documentos em que foram encontradas as palavras **foo** e **bar**.

A integração da DocRationale com essa máquina de busca é possível pois ela possui uma interface com o PHP. Assim, utilizando o PHP, a DocRationale deve preparar as consultas em linha de código no formato apropriado e enviá-las para a Swish-e. Os resultados das consultas obtidos pela Swish-e também devem ser manipulados de modo que possam ser apresentados na DocRationale.

O mecanismo definido pela Opção 3 está representado na Figura 6.7

Finalizada a descrição dessa opção, as vantagens e desvantagens de cada uma das opções apresentadas são discutidas na próxima seção.

6.7 Comparação das Opções Definidas para o Mecanismo de Recuperação de DR

A Opção 1, que utiliza o Cocoon para gerar os documentos XML e o eXist para armazená-los e buscá-los, apresenta a vantagem do Cocoon estar embutido no eXist, evitando ter que implementar um programa para gerar esses documentos. No entanto, configurar o Cocoon para estabelecer a conexão com o banco de dados, assim como usar as tecnologias envolvidas não é uma tarefa trivial. Além disso, a utilização do Cocoon não poupa o mecanismo de ter que

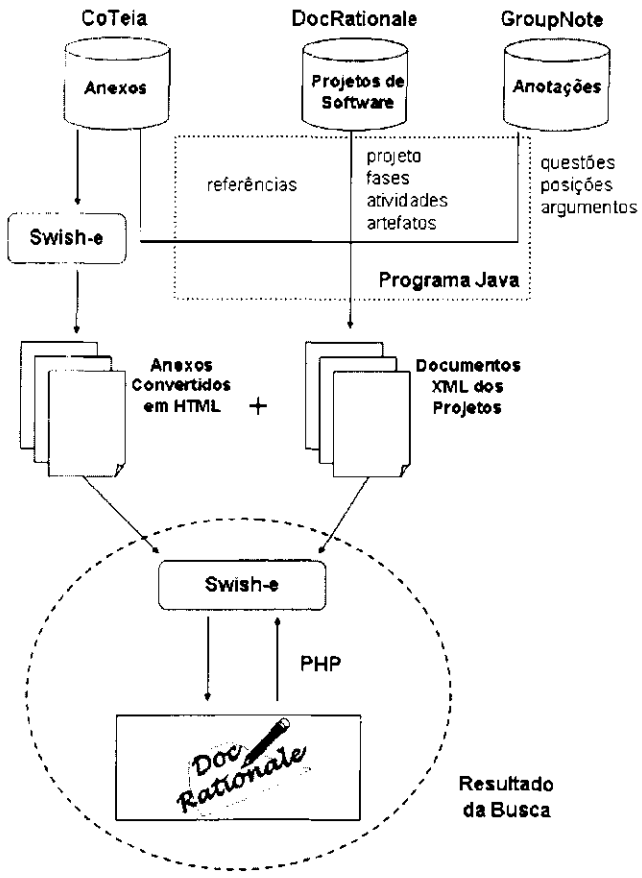

Figura 6.7: Opção para o mecanismo de recuperação de DR utilizando programa e sistema de arquivos .

possuir um programa que dispare as requisições para gerar esses documentos de tempos em tempos.

Outra vantagem da Opção 1 é que os documentos são armazenados em um banco de dados apropriado para esse tipo de documento e que além disso, oferece poderosos recursos de consulta. Entretanto, ter que utilizar diferentes ferramentas para converter os arquivos anexos e para realizar a busca nesses documentos é um ponto negativo dessa opção.

A Opção 2, que utiliza um programa para gerar os documentos XML, apesar de exigir a implementação do mesmo, é mais simples pois a implementação desse programa não é uma atividade muito complexa. Além disso, o programa implementado pode não somente gerar os documentos, mas gerá-los de tempos em tempos, poupando a implementação de dois programas. Assim como na primeira opção, a utilização do eXist é vantajosa, mas a utilização de diversas ferramentas para possibilitar a busca nos arquivos anexos ainda é um fator desfavorável.

A Opção 3, que utiliza um programa para gerar os documentos e os armazena em um sistema de arquivos, oferece as conveniências já discutidas para a geração dos documentos. Quanto à utilização do sistema de arquivos, por um lado é favorável pois possibilita que apenas uma ferramenta (Swish-e) possa ser utilizada para converter os arquivos anexos e realizar as buscas nos arquivos convertidos e nos documentos XML gerados. Por outro lado, existem dois incovenientes: primeiro, se o número de documentos XML crescer muito, esse tipo de armazenamento de torna inadequado, uma vez que nenhuma otimização na organização desses documentos é fornecida; segundo, os recursos oferecidos pela Swish-e são bem mais restritos se comparados com as possibilidades de consulta realizadas através do eXist.

A partir dessas comparações, pode-se concluir que a opção que combina programa e eXist é, teoricamente, a mais vantajosa, uma vez que gerar os documentos não é tão complicado quanto utilizar o Cocoon e ainda utiliza o eXist, que possibilita a realização de consultas mais poderosas.

6.8 A Recuperação de DR na DocRationale

A partir do momento em que uma das opções para o mecanismo de recuperação de DR for implementado na DocRationale, a recuperação de DR pode ser realizada diretamente através de um campo de busca disponível em várias telas da DocRationale ou ser refinada considerando algumas informações que sejam de interesse dos usuários. No primeiro caso, pode-se apenas realizar a busca em todos os documentos XML existentes no eXist ou em apenas um dos documentos. Na Figura 6.8 é mostrado o campo de busca disponível nas diversas telas da ferramenta.

No segundo caso, pode-se selecionar um ou mais projetos a serem pesquisados. Também é possível direcionar a busca através da escolha de outras opções como fases e atividades dos projetos, artefatos, pessoas. Ou seja, a seleção das informações possibilita ao usuário declarar quais são os projetos, as atividades, os artefatos nos quais ele deseja que a busca seja realizada. A tela na ferramenta DocRationale para a recuperação em que a busca pode ser refinada é mostrada na Figura 6.9

Os resultados obtidos a partir de quaisquer consultas, sendo elas refinadas ou não, devem ser mostrados aos usuários. Os resultados das buscas nos documentos XML podem ser tratados, por exemplo, através de transformações XSLT (*Extensible Stylesheet Language Transformer*)

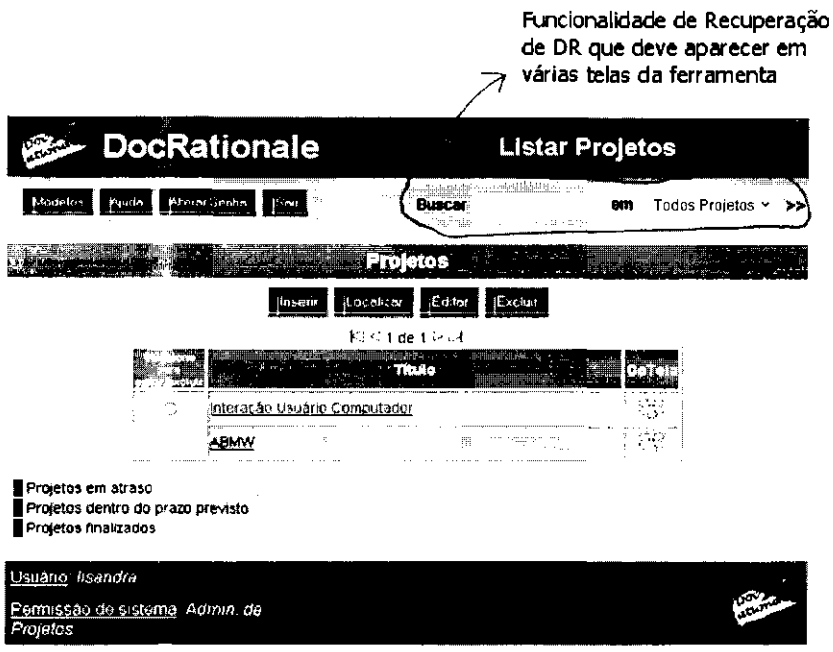

Figura 6.8: Campo de busca disponível nas telas da DocRationale.

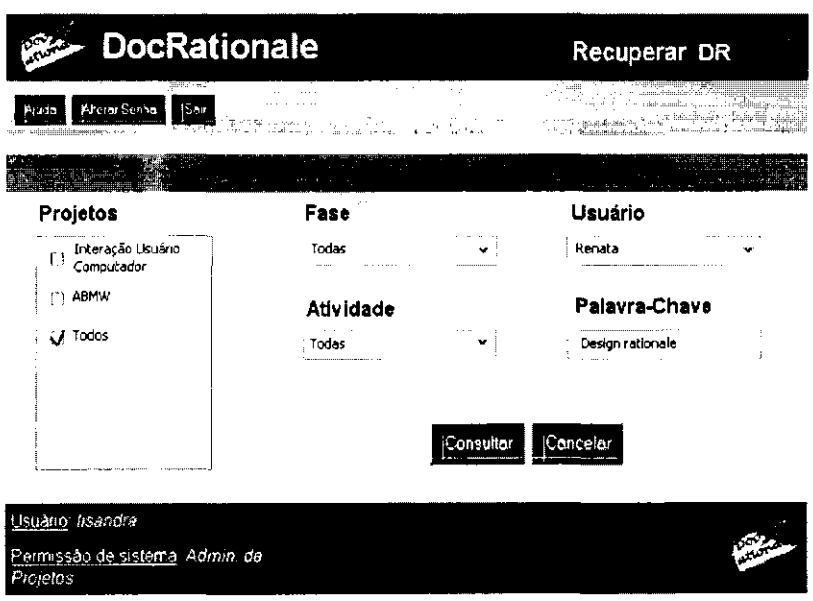

Figura 6.9: Tela para refinar a busca na DocRationale.

[W3C, 1999], de maneira que os trechos de documentos XML retornados sejam apresentados de maneira simples ao usuário.

Quanto aos arquivos convertidos em documentos HTML, deve-se apresentar os *links* para

esses documentos, como acontece normalmente quando se utiliza uma máquina de busca para recuperar informações.

Dessa forma, a apresentação dos resultados das buscas permite auxiliar na interpretação e visualização dos DRs a serem explorados. Além disso, busca-se com a utilização do mecanismo, recuperar os DRs obtidos de maneira mais eficiente e ágil. No entanto, deve-se ressaltar que esse mecanismo de recuperação não poupa o usuário de ler as informações recuperadas para encontrar, ou não, a informação que realmente deseja.

A exploração a partir dessas informações pode contribuir para que o usuário tenha conhecimento de outras informações também relacionadas, e para as quais não teria condições de ao simplesmente navegar, procurar por elas.

6.9 Considerações Finais

Na DocRationale, apenas a navegação pode ser utilizada como estratégia de recuperação do DR armazenado na ferramenta. Nesse sentido, buscou-se melhorar a recuperação e exploração desse DR, determinando opções para um mecanismo de recuperação que realizasse buscas mais eficientes e rápidas.

Dentre as abordagens de recuperação de DR estudadas, a abordagem escolhida para a ferramenta DocRationale foi a recuperação baseada em consulta através de busca por palavras e frases. O objetivo é integrar os dados armazenados nas diferentes bases de dados (DocRationale, GroupNote e Coteia), agregando-os em um único documento XML para cada projeto. Uma vez criados os documentos XML de cada projeto, palavras e frases devem ser buscadas nesses documentos com o propósito de promover a recuperação dos DRs inicialmente armazenados na DocRationale. Os arquivos anexos, relacionados aos artefatos dos projetos, também devem ser considerados na recuperação de DR. Para isso, alguns formatos desses arquivos (Microsoft Word, PDF, texto e RTF) são convertidos, e buscados por ferramentas de busca.

Para criar os documentos XML duas alternativas foram consideradas: utilizar o Cocoon através de documentos XSP ou implementar um programa, por exemplo, em Java. Em ambos os casos, primeiro as consultas nos bancos de dados são realizadas e depois os documentos são gerados, seguindo a estrutura do documento XML Schema definido para eles. De maneira geral, criar os documentos utilizando programas é mais simples, uma vez que configurar o Cocoon para gerar os documentos é uma tarefa bem complexa. Além disso, também é preciso

implementar um programa para disparar as requisições no Cocoon para que esses documentos sejam gerados regularmente.

Para armazenar e buscar os documentos XML também foram considerardas duas alternativas: utilizar o eXist para armazenar e realizar as buscas nos documentos XML ou armazenar os documentos XML em um sistema de arquivos e realizar as buscas através de uma ferramenta de busca para esse tipo de documento. Apesar do uso de sistemas de arquivo ser mais trivial, o eXist oferece as vantagens de gerenciar os documentos XML, o que não acontece no outro caso, e também de possuir recursos mais poderosos para a busca nos mesmos.

Além das mencionadas anteriormente, outras tecnologias e ferramentas de suporte à recuperação de DR foram utilizadas para compor três diferentes soluções para definir esse mecanismo. Na primeira opção, os documentos são gerados pelo Cocoon e armazenados e buscados pelo eXist. Os arquivos anexos são convertidos por ferramentas de conversão (wvWare, pdftohtml, txt2html, rtf-converter) e buscados por ferramentas de busca como mnoGoSearch ou DataparkSearch Engine. Na segunda opção, a única diferença em relação à opção anterior é que os documentos XML são gerados por um programa. Já na terceira opção, os documentos continuam sendo gerados por um programa, mas eles são armazenados em sistemas de arquivo e a busca realizada pela ferramenta de busca Swish-e. Essa ferramenta além de efetuar buscas em documentos XML, também converte diversos tipos de arquivos em HTML e realiza buscas nesses arquivos, sendo portanto também utilizada para a busca nos arquivos convertidos.

Comparativamente, as duas primeiras opções têm a desvantagem de utilizar ferramentas diferentes para converter os arquivos e realizar buscas nos mesmos, o que não acontece na terceira opção que utiliza a mesma ferramenta para realizar as duas ações.

Aparentemente, utilizando apenas um julgamento teórico, a segunda opção é a melhor dentre as três. No entanto, nenhuma dessas opções foi efetivamente implementada na ferramenta, de maneira que isso pudesse ser comprovado ou rebatido.

Na ferramenta DocRationale, a partir do momento em que alguma dessas opções for implementada, a recuperação de DR poderá ser realizada diretamente através de um campo de busca disponível em várias telas da ferramenta ou ser refinada considerando algumas informações que sejam de interesse dos usuários. No entanto, em qualquer opção os resultados das buscas, tanto nos documentos XML quanto nos arquivos convertidos, devem ser manipulados para que possam ser apresentados de maneira simples ao usuário (como uma lista de *links*).

De maneira geral, busca-se com a utilização do mecanismo definido, recuperar os DRs obtidos de maneira mais objetiva e rápida. No entanto, deve-se ressaltar que esse mecanismo de recuperação não poupa o usuário de ler as informações recuperadas para encontrar, ou não, a informação que realmente deseja.

CAPÍTULO

7

Conclusões

7.1 Considerações Iniciais

Notadamente, durante o processo de software, as informações relacionadas às decisões de projeto são muito úteis para outros projetos, uma vez que erros podem ser evitados e alternativas anteriormente consideradas podem ser mais bem reaproveitadas. Além disso, soluções discutidas e adotadas em um projeto podem ser relevantes a outros. A ferramenta DocRationale foi desenvolvida nesse contexto, pois possibilita o armazenamento de *Design Rationale* (DR), que se constitui de uma base de informações relacionadas às decisões tomadas e também às razões que propiciaram cada decisão, incluindo suas justificativas, alternativas consideradas, assim como o raciocínio empregado no desenvolvimento de um projeto.

No entanto, a DocRationale é uma ferramenta *Web* que ainda possui deficiências. O objetivo deste trabalho foi evoluir a DocRationale, estudando-a, assim como o contexto no qual ela se encontra: DR. Algumas melhorias foram implementadas e outras investigadas.

7.2 Contribuições

Design Rationale é a descrição das razões que justificam o projeto resultante. Utilizar DR oferece vantagens como: auxiliar a compreensão, a manutenção de um projeto e a comunicação da equipe. Para isso, entretanto, é necessário definir que perspectiva (argumentação, comunicação e documentação) e modelo de representação de DR adotar para possibilitar a sua efetiva captura e recuperação.

A recuperação de DR representou grande parte da investigação para o mecanismo de recuperação proposto neste trabalho, sendo realizada a revisão da literatura sobre a recuperação, tanto na *Web* como nos sistemas de DR. Atualmente, a recuperação de informação é extremamente importante e, para os usuários, encontrar informações relevantes é um dos principais pontos considerados. O desenvolvimento de diferentes tecnologias de recuperação de informação na *Web* foi motivado tanto pela chegada da *Web* assim como pela explosão das páginas nela localizadas. Navegação, máquinas de busca, *metasearchengines*, dentre outras, foram as tecnologias estudadas.

De maneira semelhante às tecnologias de recuperação de informação na *Web*, as abordagens adotadas nos sistemas de DR para busca de informações também foram destacadas. Essas abordagens foram: navegação, uso de *triggers*, recuperação baseada em consulta e estratégias híbridas, estudadas de maneira a oferecer suporte para a escolha de uma delas para ser utilizada no novo mecanismo de recuperação de DR proposto para a ferramenta DocRationale.

A DocRationale é uma ferramenta *Web* para a captura e recuperação tanto de dados de projetos, assim como dos DRs relacionados aos artefatos desses projetos. A ferramenta oferece diversas funcionalidades como criação e edição de páginas *Web*, *upload* de arquivos e uso de anotações. No entanto, atualmente utiliza apenas a abordagem de navegação para a recuperação do DR armazenado. Essa é uma das deficiências da DocRationale.

Uma análise de uso da ferramenta também apontou outras deficiências como não ser facilmente compreendida e não ter suas funcionalidades descobertas de maneira simples. Problemas de interação também foram indicados. Como o objetivo de minimizar essas deficiências, foram implementadas melhorias. A primeira delas foi elaborar documentos de ajuda para facilitar a utilização e o entendimento da ferramenta. A segunda foi acrescentar a funcionalidade de inserção automática dos projetos a partir de documentos XML.

Outra melhoria sugerida para a DocRationale foi o aperfeiçoamento na representação de DR. Essa nova funcionalidade permitiria oferecer outros modelos, como IBIS, QOC, PHI e DRL, para estruturar o DR capturado. Para isso, foram elaborados documentos XML Schema para definir a estrutura da representação linear desses modelos. A aquisição do DR seria realizada de maneira semelhante à inserção automática dos projetos: via documentos XML.

No entanto, a principal deficiência/limitação da DocRationale ainda é oferecer somente a navegação para recuperar o DR porque, conforme a quantidade de projetos aumenta, essa estratégia torna a recuperação do DR bastante custosa para o usuário. Procurando viabilizar uma nova estratégia, tecnologias *Web* que poderiam oferecer suporte a essa nova estratégia foram também estudadas. Além de oferecerem funcionalidades desejáveis ao mecanismo de recuperação, essas ferramentas eram integráveis à DocRationale.

A linguagem XML foi uma das tecnologias analisadas, uma vez que documentos XML são legíveis para os computadores, sendo armazenados, processados, buscados e exibidos.

Outra tecnologia de suporte à recuperação de DR estudada foi o Cocoon, um *framework* de desenvolvimento *Web* construído ao redor do conceito de desenvolvimento *Web* baseado em componentes, implementado em torno da nação de *pipelines*.

O eXist foi uma alternativa analisada para prover o armazenamento de documentos XML. Além de gerenciar os documentos XML, o eXist também oferece poderosos recursos para a busca nesses documentos.

Quatro ferramentas de busca em documentos HTML também foram consideradas. Todas essas ferramentas têm código aberto ou são livres, isto é, são gartuitas mesmo com código fonte não disponível. Além disso, oferecem versão para Linux e realizam buscas por palavras ou frases. As características particulares de cada uma delas também foram apresentadas neste trabalho.

Para o mecanismo de recuperação de DR proposto para a DocRationale, a abordagem escolhida foi a recuperação baseada em consulta através de busca por palavras e frases. Os dados armazenados nas diferentes bases de dados (DocRationale, GroupNote e Coteia) devem ser agregados em um único documento XML para cada projeto. Criados os documentos XML de cada projeto, palavras e frases devem ser buscadas nesses documentos com o propósito de promover a recuperação dos DRs inicialmente armazenados na DocRationale. Os arquivos anexos, relacionados aos artefatos dos projetos, também devem ser considerados na recuperação de

DR. Para isso, alguns formatos desses arquivos (Microsoft Word, PDF, texto e RTF) devem ser convertidos, e buscados por ferramentas de busca.

Para gerar os documentos XML foram consideradas duas alternativas: utilizar o Cocoon ou implementar um programa. Em ambos os casos, primeiro as consultas nos bancos de dados são realizadas e depois os documentos são gerados, seguindo a estrutura do documento XML Schema definido. De maneira geral, criar os documentos utilizando programas é mais simples, pois configurar o Cocoon ainda é uma tarefa complexa. Além disso, também é preciso implementar um programa para disparar as requisições no Cocoon para que os documentos sejam gerados de tempos em tempos.

Para armazenar e buscar os documentos XML também foram considerardas duas alternativas: utilizar o eXist para armazenar e realizar as buscas nos documentos XML ou armazenar os documentos XML em um sistema de arquivos e realizar as buscas através de uma ferramenta de busca. Apesar do uso de sistemas de arquivo ser mais trivial, o eXist oferece vantagens como o gerenciamento dos documentos XML e o oferecimento de recursos poderosos para a busca.

Além das já mencionadas, outras tecnologias e ferramentas de suporte à recuperação de DR foram utilizadas para compor três diferentes opções para o mecanismo. Na primeira opção, os documentos são gerados pelo Cocoon e armazenados e buscados pelo eXist. Os arquivos anexos são convertidos por ferramentas de conversão (wvWare, pdftohtml, txt2html, rtf-converter) e buscados por ferramentas de busca como mnoGoSearch ou DataparkSearch Engine. Na segunda opção, a única diferença em relação à opção anterior é que os documentos XML são gerados por um programa. Já na terceira opção, os documentos continuam sendo gerados por um programa, mas eles são armazenados em sistemas de arquivo e a busca realizada pela ferramenta de busca Swish-e. Essa ferramenta além de efetuar buscas em documentos XML, também converte diversos tipos de arquivos em HTML e realiza buscas nesses arquivos, sendo portanto também utilizada para a busca nos arquivos convertidos.

Comparativamente, as duas primeiras opções têm a desvantagem de utilizar ferramentas diferentes para converter os arquivos e realizar buscas nos mesmos, o que não acontece na terceira opção que utiliza a mesma ferramenta para realizar as duas ações.

Aparentemente, a segunda opção é a melhor dentre as três. No entanto, essas opções precisam ser completamente implementadas e validadas com experimentos de execução, manutenção e evolução da ferramenta. Pode-se prever que a primeira opção, adotando o Cocoon, viabiliza uma manutenção facilitada.

Na ferramenta DocRationale, a partir do momento em que alguma dessas opções for implementada, a recuperação de DR poderá ser realizada diretamente através de um campo de busca disponível em várias telas da ferramenta ou ser refinada considerando algumas informações que sejam de interesse dos usuários. No entanto, em qualquer opção os resultados das buscas, tanto nos documentos XML quanto nos arquivos convertidos, devem ser manipulados para que possam ser apresentados ao usuário como uma lista de *links*.

De maneira geral, busca-se com a utilização do mecanismo proposto, recuperar os DRs obtidos de maneira mais objetiva e rápida. No entanto, deve-se ressaltar que esse mecanismo de recuperação não poupa o usuário de ler as informações recuperadas para encontrar, ou não, a informação que realmente deseja.

7.3 Trabalhos Futuros

Como continuidade deste trabalho, deve-se implementar na DocRationale as três opções, descritas como soluções para o mecanismo de recuperação de DR, visando a experimentação. O objetivo é obter dados reais de tempo, espaço, qualidade dos resultados para realizar uma análise efetiva das opções consideradas. Além disso, deve-se evoluir a ferramenta para possibilitar que outros modelos de representação, utilizando os documentos XML definidos, possam ser considerados na captura do DR.

Como atualmente a DocRationale está sendo integrada ao **iClass** [Lara & Fortes, 2004] (sistema para a captura e acesso de sessões) para facilitar a captura de DR, outro trabalho futuro pode ser a exploração de mecanismos de recuperação de DR via computação ubíqua.

Outro trabalho seria aprimorar a DocRationale para oferecer suporte à inclusão de diferentes metadados de maneira a possibilitar a recuperação de DR de forma mais flexível, considerando um trabalho de doutorado [Pansanato, 2004] que investiga a utilização de metadados para auxiliar a navegação em documentos *Web*.

Referências Bibliográficas

ANDERSEN, K. *txt2html*. Disponível em: <http://txt2html.sourceforge.net>. Acesso em:15/12/2004, 2001.

ARRUDA, C. R. E.; IZEKI, C. A.; PIMENTEL, M. G. C. Coteia: Uma ferramenta colaborativa de edição baseada na web. In: *Anais do Workshop de Ferramentas e Aplicações do VII SBMIDIA*, Fortaleza, Brasil, 2002, p. 371–375.

ASPSEEK.ORG *Advanced internet search engine*. Disponível em: <http://www.aspseek.org>. Acesso em:13/01/2005, 2004.

BANARES-ALCANTARA, R.; KING, J. M. P. Design support systems for process engineering - design rationale as a requirement for effective support. *Computers and Chemical Engineering*, v. 21, n. 3, p. 263–276, 1997.

BORGES, L. S.; FALBO, R. A. Managing software process knowledge. In: *Proceeding of the International Conference on Computer Science, Software Engineering, Information Technologies, e-Business and Applications - (CSITea'2002)*, Foz do Iguaçu, Brasil, 2002, p. 227–232.

BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, C. M. Extensible markup language (xml). *World Wide Web Journal - Principles, Tools and Techniques*, v. 2, n. 4, p. 29–66, 1997.

BRUSILOVSKY, P. Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction*, v. 6, n. 2–3, p. 87–129, 1996.

BRUSILOVSKY, P. Adaptive hypermedia. *User Modeling and User Adapted Interaction*, v. 11, n. 1, p. 87–110, 2001.

BURGE, J.; BROWN, D. C. Reasoning with design rationale. In: *Proceedings of the Sixth International Conference on Artificial Intelligence in Design (AID'00)*, Worcester, EUA, 2000.

BURGE, J.; BROWN, D. C. Design rationale for software maintenance. In: *Proceedings of the Sixteenth IEEE International Conference on Automated Software Engineering*, San Diego, EUA, 2001, p. 433.

CERI, S.; COMAI, S.; DAMIANI, E.; FRATERNALI, P.; PARABOSCHI, S.; TANCA, L. Xml-gl: A graphical language for querying and restructuring xml documents. *Computer Networks*, v. 31, n. 11–16, p. 1171–1187, 1999.

CHANDRASEKARAN, B.; IWASAKI, Y. Functional representation as design rationale. *IEEE Computer*, v. 26, n. 1, p. 48–53, 1993.

CHANG, J. An xml document retrieval system supporting structure and content-based queries. In: ARISAWA, H.; KAMBAYASHI, Y., eds. *Conceptual Modeling for New Information Systems Technologies*, Springer, p. 320–333, 2001.

CHUNG, P. W. H.; GOODWIN, R. An integrated approach to representing and accessing design rationale. *Engineering Applications in Artificial Intelligence*, v. 11, n. 1, p. 149 159, 1998.

COCOON - THE APACHE SOFTWARE FOUNDATION. *Cocoon 2.1* Disponível em: <http://www.w3.org/TR/xmlschema-0/>. Acesso em:15/12/2004, 2004.

CONKLIN, E. J.; YAKEMOVIC, K. C. B. A process-oriented approach to design rationale. *Human Computer Interaction*, v. 6, n. 3–4, p. 357–391, 1991.

CONKLIN, J.; BEGEMAN, M. L. gibis: A hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, v. 6, n. 4, p. 303–331, 1988.

CROFT, W. B. What do people want from information retrieval? *D-Lib Magazine*, disponível em: <http://www.dlib.org/dlib/november95/11.croft.html>. Acesso em:29/10/2004, 1995.

EXIST DB.ORG *exist*. Disponível em: <http://exist.sourceforge.net>. Acesso em:13/01/2005, 2000.

DPSEARCH *Dataparksearch engine*. Disponível em: <http://www.dataparksearch.org>. Acesso em:13/01/2005, 2004.

DREILINGER, D.; HOWE, A. E. Experiences with selecting search engines using metasearch. *ACM Transactions on Information Systems*, v. 15, n. 3, p. 195–222, 1997.

FISCHER, G.; LEMKE, A. C.; MCCALL, R. Making argumentation serve design. *Human Computer Interaction*, v. 6, n. 3–4, p. 393–419, 1991.

FISCHER, G.; MCCALL, R. Janus: Integrating hypertext with a knowledge-based design environment. In: *Proceedings of the Second Annual ACM Conference on Hypertext*, Pittsburgh, EUA, 1989, p. 105–117.

FLICKNER, M.; SAWHNEY, H.; NIBLACK, W.; ASHLEY, J.; HUANG, Q.; DOM, B.; GORKANI, M.; HAFNER, J.; LEE, D.; PETKOVIC, D.; STEELE, D.; YANKER, P. Query by image and video content: The qbic system. In: MAYBURY, M. T., ed. *Intelligent Multimedia Information Retrieval*, MIT Press, p. 7–22, 1997.

FORD, G. *rtf-converter*. Disponível em: <http://freshmeat.net/projects/rtf-converter>. Acesso em:15/12/2004, 2003.

FRANCISCO, S.; IZEKI, C. A.; PAIVA, D. M. B.; FORTES, R. P. M. Um sistema de apoio à utilização de design rationale de artefatos de software. In: *Anais da XXIX Conferência Latinoamericana de Informática (CLEI 2003)*, La Paz, Bolivia, 2003.

FRANCISCO, S. D. Docrationale: Uma ferramenta para suporte a design rationale de artefatos de software. Dissertação de Mestrado. Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo, 2004.

FUHR, N. Xml information retrieval and information extraction. In: FRANKE, F.; NAKHAEIZADEH, G.; RENZ, I., eds. *Text Mining: Theoretical Aspects and Applications*, Physica-Verlag, p. 21–32, 2003.

FUHR, N.; GROJOHANN, K. Xirql: A query language for information retrieval in xml documents. In: *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR'01)*, 2001, p. 172–180.

FUMAGALLI, L. C.; FORTES, R. P. M. Recuperação de design rationale de documentos de software. In: *Anais do VII Workshop de Teses em Engenharia de Software - XVII Simpósio Brasileiro de Engenharia de Software - (SBES'2003)*, Manaus, Brasil, 2003.

FUMAGALLI, L. C.; FORTES, R. P. M. Exploração de design rationale de artefatos de software na web - um mecanismo de busca em documentos xml. In: *Anais da XXIX Conferência Latinoamericana de Informática (CLEI 2003)*, Arequipa, Peru, 2004.

FUMAGALLI, L. C.; PANSANATO, L. T. E.; FORTES, R. P. M. Documentation process in interactive systems - a case study to abstract its structure. In: *Proceedings of the Second International Information and Telecommunication Technologies Symposium (I2TS'2003)*, Florianópolis, Brasil, 2003.

GARCIA, A. C. B.; SOUZA, C. S. Add+: Including rhetorical structures in active documents. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, v. 11, n. 2, p. 109–124, 1997.

GARZA, J. M. D. L.; ALCANTARA, P. T. Using parameter dependency network to represent design rationale. *Journal of Computing in Civil Engineering*, v. 11, n. 2, p. 102–112, 1997.

GOLDFARB, C. F. *The xml handbook*. 1 ed. Prentice Hall, 639 p., 1998.

GRAVANO, L.; GARCÍA-MOLINA, H.; TOMASIC, A. Gloss: Text-source discovery over the internet. *ACM Transactions on Database Systems*, v. 24, n. 2, p. 229–264, 1999.

GRUBER, T. R.; RUSSELL, D. M. Design knowledge and design rationale: A framework for representation, capture and use. Technical Report KSL 90-45. Knowledge System Laboratory, 1991.

GRUBER, T. R.; RUSSELL, D. M. Derivation and use of design rationale information as expressed by designers. Technical Report KSL 92-64. Knowledge System Laboratory, 1992.

HALASZ, F. G. Reflections on notecards: Seven issues for the next generation of hypermedia systems. *ACM Journal on Computer Documentation*, v. 25, n. 3, p. 71–87, 2001.

HENZINGER, M.; HEYDON, A.; MITZENMACHER, M.; NAJORK, M. Measuring index quality using randon walks on the web. In: *Proceedings of the Eighth International World Wide Web Conference (WWW'99)*, Toronto, Canadá, 1999, p. 213–225.

HU, W. C.; CHEN, Y.; SCHMALZ, M. S.; RITTER, G. X. An overview of world wide web search tecnologies. In: *Proceedings of the Fifth World Multi-Conference on System, Cybernetics and Informatics (SCI2001)*, Orlando, EUA, 2001, p. 356–361.

INGWERSEN, P. *Information retrieval interaction*. Taylor Graham Publishing, 246 p., 1992.

ISENMANN, S.; REUTER, W. D. Ibis - a convincing concept... but a lousy instrument? In: *Proceedings of the Conference on Design Interactive Systems: Processes, Practices, Methods and Techniques*, 1997, p. 163–172.

IZEKI, C. A.; ARRUDA, C. R. E.; PIMENTEL, M. G. C. An xml-based infrastructure supporting collaborative annotations as first-class hyperdocuments. In: *Anais do VII Simpósio Brasileiro de Sistemas Multimídia e Hipermídia (SBMídia)*, Florianópolis, Brasil, 2001.

KARSENTY, L. An empirical evaluation of design rationale documents. In: *Proceedings of the Conference on Human Factors in Computing Systems*, Vancouver, Canadá, 1996, p. 150–156.

KOBAYASHI, M.; TAKEDA, K. Information retrieval on the web. *ACM Computing Surveys*, v. 32, n. 2, p. 146–151, 153–55, 160, 2000.

KONOPNICKI, D.; SHMUELI, O. W3qs: A query system for the world wide web. In: *Proceedings of the Twenty-First Conference on Very Large Databases (VLDB'95)*, Zurique, Suíça, 1995, p. 54–65.

KRUK, M. *pdftohtml*. Disponível em: <http://pdftohtml.sourceforge.net>. Acesso em:15/12/2004, 2003.

LACHOWICS, D. *wvware*. Disponível em: <http://wvware.sourceforge.net>. Acesso em:15/12/2004, 2000.

LAKSHMANAN, L. V. S.; SADRI, F.; SUBRAMANIAN, I. N. A declarative language for querying and restructuring the web. In: *Proceedings of the Sixth International Workshop on Research Issues in Data Engineering (RIDE'96)*, 1996, p. 54–65.

LAMSWEERDE, A. Requirements engineering in the year 2000: A research perspective. In: *Conference Proceedings (ICSE'00)*, Limerick, Irlanda: ACM, 2000, p. 5–19.

LANGHAM, M.; ZIEGELER, C. *Cocoon: Building xml applications*. 1 ed. New Riders Publishing, 504 p., 2001.

LARA, S. M. A.; FORTES, R. P. M. Um suporte à captura informal de design rationale para documentação de especificação de requisitos. In: *Proceedings of Webmedia/LA-WEB 2004 Joint Conference*, Ribeirão Preto, Brasil, 2004, p. 65–68.

LAWRENCE, S.; GILES, C. Searching the world wide web. *Science*, v. 280, p. 98–100, 1998.

LAWRENCE, S.; GILES, C. Searching the web: General and scientific information access. *IEEE Communications*, v. 37, n. 1, p. 116–122, 1999.

LEE, J. Sibyl: A tool for managing group design rationale. In: *Proceedings of the Conference on Computer Supported Cooperative Work*, Los Angeles, EUA, 1990, p. 79–92.

LEE, J. Design rationale systems: Understanding the issues. *IEEE Expert/Intelligent Systems and Their Applications*, v. 12, n. 3, p. 78–85, 1997.

LIAN, W.; CHEUNG, D. W.; MAMOULIS, N.; YIU, S. An efficient and scalable algorithm for clustering xml documents by structure. *IEEE Transactions on Knowledge and Data Engineering*, v. 16, n. 1, p. 82–95, 2004.

LIDSKY, D.; KWON, R.; LEGER, J.; RABINOVITCH, E. Searching the net. *PC Magazine*, v. 16, n. 21, p. 227–258, 1997.

LU, S. C. Y.; UBWADIA, F.; BURKETT, B.; CAI, J. Design rationale for collaborative engineering design in socio-technical framework, p. 12. Disponível em: <http://impact.usc.edu/cerl/Task2/task2-2-3/DesignRationale.pdf>. Acesso em:02/12/2004, 1999.

MARK, W.; TYLER, S.; MCGUIRE, J.; SCHLOSSBERG, J. Commitment-based software development. *IEEE Transactions on Software Engineering*, v. 18, n. 10, p. 870–886, 1992.

MARKKULA, M. Knowledge management in software engineering projects. In: *Proceedings of the Eleventh International Conference on Software Engineering and Knowledge Engineering (SEKE'99)*, Kaiserslautern, Alemanha, 1999, p. 20–27.

MCGRATH, S. *Xml aplicações práticas*. 3 ed. Editora Campus, 368 p., 1999.

MCKERLIE, D.; MACLEAN, A. Experience with qoc design rationale. In: *Adjunct Proceedings of the Conference on Human Factors in Computing Systems (INTERACT'93)*, Amsterdã, Holanda, 1993a, p. 213–214.

MCKERLIE, D.; MACLEAN, A. Qoc in action: Using dr to support design. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (INTERCHI'93)*, Amsterdã, Holanda, 1993b, p. 519.

MEIER, W. exist: An open source native xml database. In: *Revised Papers from the Web and Database-Related Workshops on Web, Web Services and Database (NODe'2002)*, Erfurt, Alemanha, 2002.

MENDELZON, A. O.; MIHAILA, G.; MILO, T. Querying the world wide web. *International Journal on Digital Libraries*, v. 1, n. 1, p. 54-67, 1997.

MNOGOSEARCH.ORG *mnogosearch*. Disponível em: <http://www.mnogosearch.org>. Acesso em:13/01/2005, 2004.

MONK, S.; SOMMERVILLE, I.; PENDARIES, J. M.; DURIN, B. Supporting design rationale for system evolution. In: *Proceedings of the Fifth European Software Engineering Conference*, Barcelona, Espanha, 1995, p. 307–323.

MORAN, T. P.; CARROLL, J. M. *Design rationale: Concepts, techniques and use (computers, cognition and work)*. 2 ed. Lawrence Erlbaum Associates, 496 p., 1996.

NIELSEN, J. *Multimedia and hypertext: The internet and beyond*. AP Professional, 480 p., 1995.

PANSANATO, L. T. E. Navegação em repositórios de informação baseados na web: Metadados como suporte a passos de navegação em uma estratégia de orientação. Monografia de Qualificação de Doutorado. Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo, 2004.

PEÑA-MORA, F.; SRIRAM, D.; LOGCHER, R. Design rationale for computer-supported conflict mitigation. *ASCE Journal of Computing in Civil Engineering*, v. 9, n. 1, p. 57–72, 1995.

PRESSMAN, R. S. *Software engineering: A practitioner's approach*. 5 ed. McGraw Hill, 915 p., 2000.

RAMESH, B.; SENGUPTA, K. Multimedia in a design rationale decision support system. *Decision Support Systems*, v. 15, n. 3, p. 181–196, 1995.

REGLI, W. C.; HU, X.; ANTWOOD, M.; SUN, W. A survey of design rationale systems: Approaches, representation, capture and retrieval. *Engineering Computers: An Int'l Journal for Simulation-Based Engineering*, v. 16, p. 209–235, 2000.

SELBERG, E.; ETZIONI, O. The metacrawler architecture for resource aggregation on the web. *IEEE Expert*, , n. January–February, p. 11–14, 1997.

SHELDON, M. A.; DUDA, A.; WEISS, R.; GIFFORD, D. K. Discover: A resource discovery system based on content routing. *Computer Networks and ISDN Systems*, v. 27, n. 6, p. 953–972, 1995.

SHIPMAN, F.; MCCALL, R. Integrating different perspectives on design rationale: Supporting the emergence of design rationale from design communication. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, v. 11, n. 2, p. 141–154, 1997.

SHUM, S. B.; HAMMOND, N. Argumentation-based design rationale: What use that what cost? *International Journal of Human-Computer Studies*, v. 40, n. 4, p. 603–652, 1994.

SMITH, J. R.; CHANG, S. F. Visualseek: A fully automated content-based image query system. In: *Proceedings of the Fourth ACM International Conference on Multimedia*, 1996, p. 23–41.

SOUZA, C.; SANTOS, D. B.; DIAS, K. L.; WAINER, J. A. Model and tool for semi-automatic recording of design rationale in software diagrams. In: *Proceedings of the String Processing and Information Retrieval Symposium and International Workshop on Groupware (SPIRE'99)*, Cancún, México, 1999, p. 306.

SWISHE.ORG *Simple web indexing system for humans - enhanced.* Disponível em: <http://www.swish-e.org>. Acesso em:13/01/2005, 2004.

TREVOR, B.; WEIPPL, E.; WINIWARTER, W. A modern approach to searching the world wide web: Ranking pages by inference over content. In: *Proceedings of the Fourteenth International Conference Applications of Prolog*, Tóquio, Japão, 2001, p. 316–330.

W3C - WORLD WIDE WEB CONSORTIUM. *XSL Transformations (XSLT) 1.0.* 1 ed. W3C Recommendation, disponível em: <http://www.w3.org/TR/xslt>. Acesso em:16/01/2005, 1999.

W3C - WORLD WIDE WEB CONSORTIUM. *Extensible Markup Language (XML) 1.0.* 2 ed. W3C Recommendation, disponível em: <http://www.w3.org/TR/2000/REC-xml-20001006>. Acesso em:02/05/2004, 2001a.

W3C - WORLD WIDE WEB CONSORTIUM. *XML Schema Definition (XSD) 1.1.* 2 ed. W3C Recommendation, disponível em: <http://www.w3.org/TR/xmlschema-0/>. Acesso em:15/12/2004, 2001b.

WANGENHEIM, C. G.; ALTHOFFL, K. D.; BARCIA, R. M. Intelligent retrieval of software engineering experienceware. In: *Proceedings of the Eleventh International Conference on Software Engineering and Knowledge Engineering (SEKE'99)*, Kaiserslautern, Alemanha, 1999, p. 128–135.

WIEGERAAD, S. Development of a design history information system: Capturing and re-using the knowledge behind the product. *Tese de Doutorado. Technische Universiteit Eindhoven*, p. 164, disponível em: <http://alexandria.tue.nl/extra3/proefschrift/boeken/9902517.pdf>. Acesso em:02/12/2004, 1999.

XSP. *Extensible Server Pages.* Disponível em: <http://cocoon.apache.org/2.1/userdocs/xsp/index.html>. Acesso em:13/01/2005, 2004.

XU, G.; COCKBURN, A.; MCKENZIE, B. Lost on the web: An introduction to web navigation research. In: *Proceedings of the Fourth NewZealand Computers Science Students Conference (NZCSSC'01)*, Christchurch, New Zealand, 2001.

YOON, J. P.; RAGHAVAN, V.; CHAKILAM, V.; KERSCHBERG, L. Bitcube: A three-dimensional bitmap indexing for xml documents. *Journal of Intelligent Information Systems*, v. 17, n. 2–3, p. 241–254, 2001.

APÊNDICE

A

Questionário sobre Facilidade de Uso da DocRationale

As perguntas feitas aos alunos sobre a DocRationale e as possíveis respostas a cada uma delas, especialmente em relação à facilidade de uso da ferramenta, são apresentadas a seguir.

É fácil aprender a usar a DocRationale?
( ) Não, não consegui aprender a utilizar o sistema.
( ) Não, mas aprendi com muitas dificuldades.
( ) Parcialmente.
( ) Sim, mas tive algumas dificuldades no início.
( ) Sim, e não tive nenhuma dificuldade.

É fácil entender o conceito lógico da ferramenta DocRationale e sua aplicabilidade?
( ) Não, é muito difícil entender.
( ) Parcialmente, pois é possível entender somente após algum tempo de uso.
( ) Sim, é possível entender logo no início da interação.

A ferramenta DocRationale é fácil de utilizar e controlar?
( ) Não, é muito complicado.
( ) Parcialmente, pois não foi fácil utilizar o sistema para cumprir algumas das tarefas. (Neste caso especificar tais tarefas).
( ) Sim, é muito fácil.

A interface (como um todo) auxilia na aplicação da ferramenta DocRationale?
( ) Não.
( ) Parcialmente, pois os recursos oferecidos não são satisfatórios.
( ) Sim, os recursos oferecidos são satisfatórios e suficientes.

A interface auxilia o usuário em uma realização rápida de suas finalidades (atividades/objetivos)?
( ) Não, o realização é demorada.
( ) Apresenta nível tolerável.
( ) Sim, a realização é rápida.

A interface auxilia o usuário em uma realização fácil de suas finalidades (atividades/objetivos)?
( ) Não, a realização é complicada.
( ) Parcialmente, pois os recursos oferecidos não são satisfatórios.
( ) Sim, o a realização é fácil.

A inserção de projetos é rápida e fácil?
( ) Não, a criação de projetos é complicada.
( ) Parcialmente, pois os recursos oferecidos não são satisfatórios.
( ) Sim, a criação de projetos é fácil.
( ) Nunca inseri projetos.

A inserção de artefatos de projeto é fácil de ser feita?
( ) Não, é complicado inserir artefatos de projetos.
( ) Parcialmente, pois os recursos oferecidos não são satisfatórios.
( ) Sim, a inserção de artefatos de projeto é fácil.
( ) Nunca inseri artefatos.

A inserção de questões de projeto é fácil de ser feita?
( ) Não, é complicado inserir questões de projetos.
( ) Parcialmente, pois os recursos oferecidos não são satisfatórios.
( ) Sim, a inserção de questões de projeto é fácil.
( ) Nunca inseri questões.

A criação de posições (respostas) relacionadas às questões é fácil de ser feita?
( ) Não, é complicado criar as posições das questões de projeto.
( ) Parcialmente, pois os recursos oferecidos não são satisfatórios.
( ) Sim, a criação de posições das questões de projeto é fácil.
( ) Nunca criei posições.

A composição da equipe de projeto é fácil de ser feita?
( ) Não, é complicado compor a equipe.
( ) Parcialmente, pois os recursos oferecidos não são satisfatórios.
( ) Sim, a composição da equipe de projeto é fácil.
( ) Nunca compus uma equipe.

Encontrar um projeto específico é fácil de ser feito?
( ) Não, é complicado encontrar um projeto específico.
( ) Parcialmente, pois a navegação às vezes é complexa.
( ) Sim, encontrar um projeto específico é fácil.

A funcionalidade Localizar é fácil de ser utilizada?
( ) Não, a funcionalidade não é fácil de ser utilizada.
( ) Parcialmente, pois é necessário saber exatamente o que se deseja localizar.
( ) Sim, a funcionalidade é fácil de ser encontrada.
( ) Nunca usei essa funcionalidade.

A visualização de seus projetos é mostrada de forma:
( ) Péssima, não entendi nada.
( ) Ruim, só consegui entender depois de algum tempo.
( ) Boa, mas o espaço de tempo mostrado é muito curto.
( ) Muito boa, apenas pequenos detalhes precisam ser melhorados.
( ) Excelente, esta é a melhor maneira.

Se você continuou a utilizar a DocRationale mais de uma vez, qual foi o principal fator que o levou a utilizá-lo posteriormente?
( ) Necessidade
( ) Facilidades oferecidas pela ferramenta
( ) Autonomia
( ) Só usei a ferramenta uma vez
( ) Outros (especifique).

Alguma vez houve algum motivo que o levou a não utilizar a ferramenta DocRationale? (Pode ser marcada mais de uma alternativa)
( ) A ferramenta não oferece as funcionalidades que eu necessito.
( ) Esquecimento.
( ) Existem outras pessoas que podem fazer isso por mim.
( ) Acho difícil utilizar a ferramenta.
( ) Já estou acostumado a usar outras ferramentas e utilizar a DocRationale não facilita.
( ) Outros (especifique).

Você encontrou alguma dificuldade em usar a ferramenta DocRationale? Se sim, quais? (Pode ser marcada mais de uma alternativa).
( ) O caminho de navegação é complexo.
( ) A ferramenta tem funcionalidades que desconheço.
( ) O processo para criar a página onde devo fazer o upload é complexo.
( ) Fazer o upload dos arquivos é complicado.
( ) Outros (especifique).

Para que finalidades você utiliza a ferramenta DocRationale? (Pode ser marcada mais de uma alternativa).

( ) Para entrega de artefatos de projeto.
( ) Para discussão dos projetos.
( ) Como meio alternativo de armazenamento de arquivos.
( ) Outros (especifique).

Você acha que uma seção Ajuda seria necessária?
( ) Não
( ) Sim

Você acha que uma seção Busca seria necessária?
( ) Não

( ) Sim. Se sim, o que ela deveria buscar?

APÊNDICE

B

XML Schema dos Modelos de Representação de DR

Como mencionado anteriormente, o XML Schema (ou XSD) é um documento que descreve a estrutura de um documento XML. Assim, foram definidos os documentos XSD que descrevem a estrutura dos documentos XML que transcrevem a representação dos modelos IBIS, PHI, QOC e DRL. Também foi definido o XML Schema que descreve a estrutura relacionada ao modelo de representação (PHI simplificado) utilizado na DocRationale para o registro de DR.

Todos esses XML Schemas são apresentados neste apêndice.

B.1 XML Schema do Modelo IBIS

```
<?xml version="1.0" encoding="ISO-8859"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!- Declaração dos Elementos ->
<xs:element name="QUESTION_TITLE" type="xs:string"/>
<xs:element name="POSITION_TITLE" type="xs:string"/>
<xs:element name="ARGUMENT_TITLE" type="xs:string"/>

<!- Declaração do Atributo ->
<xs:attribute name="LINK" type="xs:string"/>

<!- Declaração dos Tipos e Elementos ->
<xs:complexType name="argumentType">
   <xs:sequence>
     <xs:element ref="ARGUMENT_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element name="QUESTION" type="questionType" minOccurs="0"
                 maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute ref="LINK" use="required"/>
</xs:complexType>

<xs:complexType name="positionType">
   <xs:sequence>
     <xs:element ref="POSITION_TITLE" minOccurs="1"
                 maxOccurs="1"/>
     <xs:element name="ARGUMENT" type="argumentType" minOccurs="1"
                 maxOccurs="unbounded"/>
     <xs:element name="QUESTION" type="questionType" minOccurs="0"
                 maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute ref="LINK" use="required"/>
</xs:complexType>

<xs:complexType name="questionType">
   <xs:sequence>
     <xs:element ref="QUESTION_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element name="POSITION" type="positionType" minOccurs="0"
                 maxOccurs="unbounded"/>
     <xs:element name="QUESTION" type="questionType" minOccurs="0"
                 maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute ref="LINK" use="required" default="initial"/>
</xs:complexType>

<xs:element name="IBIS">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element name="QUESTION" type="questionType" minOccurs="1"
                    maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

B.2 XML Schema do Modelo PHI

```
<?xml version="1.0" encoding="ISO-8859"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!- Declaração dos Elementos ->
<xs:element name="ISSUE_TITLE" type="xs:string"/>
<xs:element name="SUB_ISSUE_TITLE" type="xs:string"/>
<xs:element name="ANSWER_TITLE" type="xs:string"/>
<xs:element name="SUB_ANSWER_TITLE" type="xs:string"/>
<xs:element name="ARGUMENT_TITLE" type="xs:string"/>
<xs:element name="SUB_ARGUMENT_TITLE" type="xs:string"/>

<!- Declaração dos Tipos e Elementos ->
<xs:complexType name="sub_argumentType">
   <xs:sequence>
     <xs:element ref="SUB_ARGUMENT_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element name="SUB_ARGUMENT" type="sub_argumentType"
                    minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="argumentType">
   <xs:sequence>
     <xs:element ref="ARGUMENT_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element name="SUB_ARGUMENT" type="sub_argumentType"
                    minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="sub_answerType">
   <xs:sequence>
     <xs:element ref="SUB_ANSWER_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element name="ARGUMENT" type="argumentType" minOccurs="0"
                    maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="answerType">
   <xs:sequence>
     <xs:element ref="ANSWER_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element name="ARGUMENT" type="argumentType" minOccurs="0"
                    maxOccurs="unbounded"/>
     <xs:element name="SUB_ANSWER" type="sub_answerType" minOccurs="0"
                    maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="sub_issueType">
   <xs:sequence>
     <xs:element ref="SUB_ISSUE_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element name="ANSWER" type="answerType" minOccurs="0"
                    maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="issueType">
   <xs:sequence>
     <xs:element ref="ISSUE_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element name="SUB_ISSUE" type="sub_issueType" minOccurs="0"
                    maxOccurs="unbounded"/>
     <xs:element name="ANSWER" type="answerType" minOccurs="1"
                    maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>

<xs:element name="PHI">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="ISSUE" type="issueType" minOccurs="1"
                      maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>
</xs:schema>
```

B.3 XML Schema do Modelo QOC

```
<?xml version="1.0" encoding="ISO-8859"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!- Declaração dos Elementos ->
<xs:element name="QUESTION_TITLE"type="xs:string"/>
<xs:element name="OPTION_TITLE" type="xs:string"/>
<xs:element name="CRITERION_TITLE" type="xs:string"/>

<!- Declaração do Atributo ->
<xs:attribute name="LINK" type="xs:string"/>

<!- Declaração dos Tipos e Elementos ->
<xs:complexType name="criterionType">
   <xs:sequence>
     <xs:element ref="CRITERION_TITLE" minOccurs="1" maxOccurs="1"/>
   </xs:sequence>
   <xs:attribute ref="LINK" use="required"/>
</xs:complexType>

<xs:complexType name="optionType">
   <xs:sequence>
     <xs:element ref="OPTION_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element name="CRITERION" type="criterionType" minOccurs="1"
                 maxOccurs="unbounded"/>
     <xs:element name="QUESTION" type="questionType" minOccurs="0"
                 maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute ref="LINK" use="required"/>
</xs:complexType>

<xs:complexType name="questionType">
   <xs:sequence>
     <xs:element ref="QUESTION_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element name="OPTION" type="optionType" minOccurs="1"
                 maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute ref="LINK" use="required" default="initial"/>
</xs:complexType>

<xs:element name="QOC">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="QUESTION" type="questionType" minOccurs="1"
                   maxOccurs="unbounded"/>
     </xs:sequence>
```

```
    </xs:complexType>
</xs:element>
</xs:schema>
```

B.4 XML Schema do Modelo DRL

```
<?xml version="1.0" encoding="ISO-8859"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!- Declaração dos Elementos ->
<xs:element name="PROCEDURE_TITLE"type="xs:string"/>
<xs:element name="CLAIM_TITLE" type="xs:string"/>
<xs:element name="QUESTION_TITLE" type="xs:string"/>
<xs:element name="GROUP_TITLE" type="xs:string"/>
<xs:element name="ALTERNATIVE_TITLE" type="xs:string"/>
<xs:element name="GOAL_TITLE" type="xs:string"/>
<xs:element name="DECISION_PROBLEM_TITLE" type="xs:string"/>

<!- Declaração do Atributo ->
<xs:attribute name="LINK" type="xs:string"/>

<!- Declaração dos Tipos e Elementos ->
<xs:complexType name="procedureType">
   <xs:sequence>
     <xs:element ref="PROCEDURE_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element name="PROCEDURE" type="procedureType" minOccurs="0"
                  maxOccurs="unbounded"/>
     <xs:element name="CLAIM" type="claimType" minOccurs="0"
                  maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute ref="LINK" use="required"/>
</xs:complexType>

<xs:complexType name="claimType">
   <xs:sequence>
     <xs:element ref="CLAIM_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element name="CLAIM" type="claimType" minOccurs="0"
                  maxOccurs="unbounded"/>
     <xs:element name="CLAIM" type="claimType" minOccurs="0"
                  maxOccurs="unbounded"/>
     <xs:element name="QUESTION" type="questionType" minOccurs="0"
                  maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute ref="LINK" use="required"/>
</xs:complexType>

<xs:complexType name="questionType">
   <xs:sequence>
     <xs:element ref="QUESTION_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element name="GROUP" type="groupType" minOccurs="0"
     maxOccurs="unbounded"/>
     <xs:element name="CLAIM" type="claimType" minOccurs="0"
                  maxOccurs="unbounded"/>
```

```
    <xs:element name="PROCEDURE" type="procedureType" minOccurs="0"
                maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute ref="LINK" use="required"/>
</xs:complexType>

<xs:complexType name="groupType">
  <xs:sequence>
    <xs:element ref="GROUP_TITLE" minOccurs="1" maxOccurs="1"/>
    <xs:element name="CLAIM" type="claimType" minOccurs="0"
                maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute ref="LINK" use="required" default="is a member of"/>
</xs:complexType>

<xs:complexType name="alternativeType">
  <xs:sequence>
    <xs:element ref="ALTERNATIVE_TITLE" minOccurs="1" maxOccurs="1"/>
    <xs:element name="CLAIM" type="claimType" minOccurs="0"
                maxOccurs="unbounded"/>
    <xs:element name="ALTERNATIVE" type="alternativeType" minOccurs="0"
                maxOccurs="unbounded"/>
    <xs:element name="QUESTION" type="questionType" minOccurs="0"
                maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute ref="LINK" use="required"/>
</xs:complexType>

<xs:complexType name="goalType">
  <xs:sequence>
    <xs:element ref="GOAL_TITLE" minOccurs="1" maxOccurs="1"/>
    <xs:element name="GOAL" type="goalType" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="ALTERNATIVE" type="alternativeType" minOccurs="0"
                maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute ref="LINK" use="required"/>
</xs:complexType>

<xs:complexType name="decisionproblemType">
  <xs:sequence>
    <xs:element ref="DECISION_PROBLEM_TITLE" minOccurs="1" maxOccurs="1"/>
    <xs:element name="DECISION_PROBLEM" type="decisionproblemType"
                minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="GOAL" type="goalType" minOccurs="0"
                maxOccurs="unbounded"/>
    <xs:element name="ALTERNATIVE" type="alternativeType" minOccurs="0"
                maxOccurs="unbounded"/>
```

```
    </xs:sequence>
    <xs:attribute ref="LINK" use="required"/>
</xs:complexType>

<xs:element name="DRL">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DECISION_PROBLEM" type="decisionproblemType"
                          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

B.5 XML Schema do Modelo PHI Simplificado Utilizado na DocRationale

```
<?xml version="1.0" encoding="ISO-8859"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!- Declaração dos Elementos ->
<xs:element name="ARGUMENT_TITLE" type="xs:string"/>
<xs:element name="ARGUMENT_CONTENT" type="xs:string"/>
<xs:element name="POSITION_TITLE" type="xs:string"/>
<xs:element name="POSITION_CONTENT" type="xs:string"/>
<xs:element name="QUESTION_TITLE" type="xs:string"/>

<!- Declaração do Atributo ->
<xs:attribute name="LINK" type="xs:string"/>

<!- Declaração dos Tipos e Elementos ->
<xs:complexType name="argumentType">
   <xs:sequence>
     <xs:element ref="ARGUMENT_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element ref="ARGUMENT_CONTENT" minOccurs="1" maxOccurs="1"/>
   </xs:sequence>
   <xs:attribute name="LINK" use="required"/>
</xs:complexType>

<xs:complexType name="positionType">
   <xs:sequence>
     <xs:element ref="POSITION_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element ref="POSITION_CONTENT" minOccurs="1" maxOccurs="1"/>
     <xs:element name="ARGUMENT" type="argumentType" minOccurs="0"
                    maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="positionsType">
   <xs:sequence>
     <xs:element name="POSITION" type="positionType" minOccurs="1"
                    maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="questionType">
   <xs:sequence>
     <xs:element ref="QUESTION_TITLE" minOccurs="1" maxOccurs="1"/>
     <xs:element name="POSITIONS" type="positionsType"/>
```

```
    </xs:sequence>
</xs:complexType>

<xs:complexType name="questionsType">
    <xs:sequence>
      <xs:element name="QUESTION" type="questionType" minOccurs="1"
                    maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:element name="DOCRATIONALE">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="QUESTIONS" type="questionsType"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

APÊNDICE

C

XML Schema da DocRationale

Para integrar os dados de projeto armazenados pela DocRationale e as informações de DR registradas também pela DocRationale foi utilizado um documento XML. No entanto, a descrição da estrutura desse documento XML foi definida através do XML Schema apresentado a seguir.

```
<?xml version="1.0" encoding="ISO-8859"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!- Declaração dos Elementos ->
<xs:element name="proj_name" type="xs:string"/>
<xs:element name="proj_dt_beginning" type="xs:date"/>
<xs:element name="proj_dt_finished" type="xs:date"/>
<xs:element name="proj_manager" type="xs:string"/>
<xs:element name="pha_name" type="xs:string"/>
<xs:element name="act_name" type="xs:string"/>
<xs:element name="art_name" type="xs:string"/>
<xs:element name="art_dt_beginning" type="xs:date"/>
<xs:element name="art_dt_finished" type="xs:date"/>
<xs:element name="issue_name" type="xs:string"/>
<xs:element name="issue_dt_beginning" type="xs:date"/>
<xs:element name="issue_dt_finished" type="xs:date"/>
<xs:element name="annot_name" type="xs:string"/>
<xs:element name="content" type="xs:string"/>
<xs:element name="issue_dt_conclusion" type="xs:date"/>
<xs:element name="conclusion" type="xs:string"/>
```

```
<xs:element name="file_name" type="xs:string"/>
<xs:element name="file_dt_upload" type="xs:date"/>

<!- Declaração dos Atributos ->
<xs:attribute name="id_proj" type="xs:integer"/>
<xs:attribute name="id_pha" type="xs:integer"/>
<xs:attribute name="id_act" type="xs:integer"/>
<xs:attribute name="id_art" type="xs:integer"/>
<xs:attribute name="id_issue" type="xs:integer"/>
<xs:attribute name="id_annot" type="xs:integer"/>
<xs:attribute name="annot_type" type="xs:string"/>
<xs:attribute name="annot_user" type="xs:string"/>

<!- Declaração do elemento Issue_Conclusion que pode conter a conclusão de
uma questão ->
<xs:element name="Issue_Conclusion">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="issue_dt_conclusion" minOccurs="1" maxOccurs="1"/>
       <xs:element ref="conclusion" minOccurs="1" maxOccurs="1"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>

<!- Declaração do tipo Anotação (GroupNote) ->
<xs:complexType name="annotationType">
   <xs:sequence>
     <xs:element ref="annot_name" minOccurs="1" maxOccurs="1"/>
     <xs:element ref="content" minOccurs="1" maxOccurs="1"/>
     <xs:element name="Annotation" type="annotationType" minOccurs="0"
                    maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute ref="id_annot" use="required"/>
   <xs:attribute ref="annot_type" use="required"/>
   <xs:attribute ref="annot_user" use="required"/>
</xs:complexType>

<!- Definição do elemento Anotações que pode conter vários elementos Anota-
ção ->
<xs:element name="Annotations">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="Annotation" type="annotationType" minOccurs="1"
                       maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>
```

```
<!- Definição do elemento Questão que contém as questões inseridas na Doc-
Rationale e também todas as anotações relacionadas a essas questões ->
<xs:element name="Issue">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="issue_name" minOccurs="1" maxOccurs="1"/>
       <xs:element ref="issue_dt_beginning" minOccurs="1" maxOccurs="1"/>
       <xs:element ref="issue_dt_finished" minOccurs="1" maxOccurs="1"/>
       <xs:element ref="Annotations" minOccurs="0" maxOccurs="1"/>
       <xs:element ref="Issue_Conclusion" minOccurs="0" maxOccurs="1"/>
     </xs:sequence>
     <xs:attribute ref="id_issue" use="required"/>
   </xs:complexType>
</xs:element>

<!- Definição do elemento Questões que pode conter vários elementos Questão
->
<xs:element name="Issues">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="Issue" minOccurs="1" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>

<!- Definição do elemento Arquivo de Upload que contém os arquivos inseri-
dos na DocRationale ->
<xs:element name="File_Uploaded">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="file_name" minOccurs="1" maxOccurs="1"/>
       <xs:element ref="file_dt_upload" minOccurs="1" maxOccurs="1"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>

<!- Definição do elemento Arquivos de Upload que pode conter vários elemen-
tos Arquivo de Upload ->
<xs:element name="Files_Uploaded">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="File_Uploaded" minOccurs="1" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>

<!- Definição do elemento Artefato que contém cada um dos artefatos e tam-
bém todos as arquivos relacionados a esses artefatos inseridos na DocRatio-
```

```
nale ->
<xs:element name="Artefact">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="art_name" minOccurs="1" maxOccurs="1"/>
       <xs:element ref="art_dt_beginning" minOccurs="1" maxOccurs="1"/>
       <xs:element ref="art_dt_finished" minOccurs="1" maxOccurs="1"/>
       <xs:element ref="Issues" minOccurs="0" maxOccurs="1"/>
       <xs:element ref="Files_Uploaded" minOccurs="0" maxOccurs="1"/>
     </xs:sequence>
     <xs:attribute ref="id_art" use="required"/>
   </xs:complexType>
</xs:element>

<!- Definição do elemento Artefatos que pode conter vários elementos Arte-
fato ->
<xs:element name="Artefacts">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="Artefact" minOccurs="1" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>

<!- Definição do elemento Atividade que contém cada uma das atividades in-
seridas na DocRationale e também todos os artefatos pertencentes a essa
atividade->
<xs:element name="Activity">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="act_name" minOccurs="1" maxOccurs="1"/>
       <xs:element ref="Artefacts" minOccurs="1" maxOccurs="1"/>
     </xs:sequence>
     <xs:attribute ref="id_act" use="required"/>
   </xs:complexType>
</xs:element>

<!- Definição do elemento Atividades que pode conter vários elementos Ati-
vidade ->
<xs:element name="Activities">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="Activity" minOccurs="1" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>

<!- Definição do elemento Fase que contém todas as atividades relacionadas
```

```
a essa fase ->
<xs:element name="Phase">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="pha_name" minOccurs="1" maxOccurs="1"/>
       <xs:element ref="Activities" minOccurs="1" maxOccurs="1"/>
     </xs:sequence>
     <xs:attribute ref="id_pha" use="required"/>
   </xs:complexType>
</xs:element>

<!- Definição do elemento Fases que pode conter vários elementos Fase ->
<xs:element name="Phases">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="Phase" minOccurs="1" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>

<!- Definição do elemento Projeto que contém todas as fases do mesmo ->
<xs:element name="Project">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="proj_name" minOccurs="1" maxOccurs="1"/>
       <xs:element ref="proj_dt_beginning" minOccurs="1" maxOccurs="1"/>
       <xs:element ref="proj_dt_finished" minOccurs="1" maxOccurs="1"/>
       <xs:element ref="proj_manager" minOccurs="1" maxOccurs="1"/>
       <xs:element ref="Phases" minOccurs="1" maxOccurs="1"/>
     </xs:sequence>
     <xs:attribute ref="id_proj" use="required"/>
   </xs:complexType>
</xs:element>
</xs:schema>
```