UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Consistência de Ligações na World-Wide Web

por

LUÍS FERNANDO FORTES GARCIA

Dissertação submetida à avaliação, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. José Valdeni de Lima
Orientador

Porto Alegre, novembro de 1997.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Garcia, Luís Fernando Fortes
Consistência de Ligações na World-Wide Web / por Luís Fernando Fortes Garcia . - Porto alegre: CPGCC da UFRGS,1997.
96f.:il.

Dissertação (mestrado) - Universidade Federal do Rio Grande do Sul. Curso de Pós Graduação em Ciência da Computação, Porto alegre, BR-RS, 1997. Orientador: Lima, José Valdeni de.

1. Hipertexto. 2.Hipermídia. 3.Sistemas de Hiperdocumentos. 4. Consistência de Ligações. I. Lima, José Valdeni de. II. Título.

Agradecimentos

Gostaria de agradecer profundamente a minha esposa Letícia, a minha família e amigos pelo carinho, apoio e dedicação dispensados durante todo o tempo que julguei ser necessário para a conclusão desta dissertação.

Ao CNPq que, através de bolsa auxílio/mestrado, permitiu-me realizar este Curso de Pós-Graduação em Ciência da Computação.

Aos professores do CPGCC, em especial ao meu orientador, Prof. Dr. José Valdeni de Lima, pela preciosa orientação, dedicação - e paciência - durante todo o processo de criação e implementação desta dissertação.

Ao amigo e bolsista de Iniciação Científica Tatiano Pianezzola pelo prestimoso auxílio na implementação e verificação do protótipo.

Aos demais amigos do Grupo de Hiperdocumentos, colegas do GPGCC e funcionários do Instituto de Informática da UFRGS.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Tem-se observado que um dos maiores problemas decorrentes da expansão das ferramentas de difusão de informações em ambientes distribuídos, especialmente no modelo WWW-Internet, é a inconsistência das ligações entre os objetos envolvidos (Erro 404 - Arquivo não encontrado). Segundo [WHY 95], uma das mais importantes condições para a qualidade de uma estrutura de hiperdocumento é a manutenção da integridade e atualidade das ligações, sejam estas internas ou externas.

Sempre que há alterações na hiperbase, nas estruturas dos documentos que a compõem ou no conteúdo semântico destes, podem surgir inconsistências nas ligações. Estas podem indicar tanto um problema temporário da rede ou servidor quanto um recurso que tenha sido permanentemente movido ou excluído. Acredita-se que o percentual de requisições inválidas cresça de acordo com o tempo, com a inclusão de novos recursos e com a dinamicidade intrínseca das estruturas dos Sistemas de Hiperdocumentos Distribuídos.

Faz-se necessária, então, a adoção de modelos de Sistemas de Hiperdocumentos Distribuídos que contemplem o tratamento do problema de inconsistência de ligações desde sua concepção, através de técnicas preventivas ou então o desenvolvimento de ferramentas e técnicas para que a integridade das ligações seja garantida, preferencialmente de modo automático, para os modelos que não a apresentam nativamente.

Isto posto, este trabalho trata do problema da inconsistência de ligações nos Sistemas de Hiperdocumentos Distribuídos, com ênfase especial no modelo WWW (World-Wide Web) [BER 92].

Para tanto, são apresentados noções e conceitos, histórico, objetivos, estado da arte, arquitetura e requisitos necessários, assim como características específicas dos principais Sistemas de Hiperdocumentos Distribuídos. São tratados, também, os aspectos teóricos referentes aos problemas gerados pela inconsistência de ligações de ordem técnica, psicológica e/ou mercadológica.

São apresentadas soluções encontradas na bibliografia para os sistemas implementados, bem como é feita uma classificação destas soluções em nível de preventivas (que não permitem que o problema ocorra) e corretivas (que são utilizadas para solucionar ou minimizar o problema), podendo estas serem, ainda, de funcionamento manual, semi-automático ou automático.

A solução proposta, classificada como preventiva semi-automática, baseia-se em entrada de dados via formulário HTML e armazenamento em arquivo do tipo texto, é apresentada através de sua definição, objetivos, funcionamento e do detalhamento do protótipo, através das características de implementação e do ambiente utilizado.

Palavras-Chave: Hipertexto, Hipermídia, Sistemas de Hiperdocumentos Distribuídos, Consistência de Ligações.

TITLE: "WORLD WIDE WEB LINK CONSISTENCY"

Abstract

We have noticed that one of the biggest problems resulting from the spreading of the information diffusion tools in distributed environments (specially with the WWW-Internet model) is the inconsistency of the links between the objects involved (Error 404 - File was not found). According to [WHY 95], one of the most important conditions for the quality of an hyperdocument structure is the maintenance of the integrity of the links, whatever they may be internal or external to the hyperdocument.

Whenever there are hyperbase changes, changes on the structures of the papers that compose the hyperbase, or if there are changes on the semantic content of these papers, there may arise inconsistencies with the links. These inconsistencies may indicate a temporary net or server problem, as well as a resource that has been constantly moved or excluded. It's believed that the percentage of invalid requisites raises in accordance with time, with the introduction of new resources and with the dynamics of the structure of Distributed Hyperdocument Systems.

So, it's necessary to adopt models of Distributed Hyperdocument Systems that contemplate a treatment of the problem of the links' inconsistencies since their conception, using preventive techniques or developing tools and techniques to guarantee the integrity of the links, preferably by automatic mode to the models that originally don't have this integrity.

Thus, this work is about the problem of links' inconsistency in Distributed Hyperdocument Systems, with special emphasis on the model: WWW (World Wide Web) [BER 92].

For that, basic concepts, history, objectives, the state of the art, architecture and necessary requirements are shown., as well as specific characteristics of the main Distributed Hyperdocument Systems. Also shown are the theoretical aspects concerning the problems produced by the links inconsistency of the technical, psychological and/or market order.

Solutions found on the bibliography are shown as well as a classification of these solutions as preventives (that don't permit the problem to occur) and correctives (that are used to solve or minimize the problem) is done. These solutions may still be manual, semi-automatic or automatic.

The proposed solution, classified as preventive semiautomatic, is based on data entries using the HTML form and their storage in files in textual form. The definition, objectives, functioning and detail of a prototype are presented.

KeyWords: Hypertext, Hypermedia, Distributed Hyperdocument Systems, Consistency of Links.

1 Introdução

Um dos maiores problemas decorrentes da expansão das ferramentas de difusão de informações em ambientes distribuídos, especialmente no modelo WWW-Internet, é a inconsistência das ligações entre os objetos envolvidos (Erro 404 - Arquivo não encontrado). Segundo [WHY 95], uma das mais importantes condições para a qualidade de uma estrutura de hiperdocumento é a manutenção da integridade e atualidade das ligações, sejam estas internas ou externas a.

Sempre que há alterações na hiperbase, nas estruturas dos documentos que a compõem, ou no conteúdo semântico destes, podem surgir inconsistências nas ligações. Estas podem indicar tanto um problema temporário da rede ou servidor quanto um recurso que tenha sido permanentemente movido ou excluído. Acredita-se que o percentual de requisições inválidas cresça de acordo com o tempo, com a inclusão de novos recursos e com a dinamicidade intrínseca das estruturas dos Sistemas de Hiperdocumentos Distribuídos.

Faz-se necessária, então, a adoção de modelos de Sistemas de Hiperdocumentos Distribuídos que contemplem o tratamento do problema de inconsistência de ligações desde sua concepção, através de técnicas preventivas ou, então, o desenvolvimento de ferramentas e técnicas para que a integridade das ligações seja garantida, preferencialmente de modo automático, para os modelos que não a apresentam nativamente.

Desta forma, o objetivo deste trabalho contempla a pesquisa e classificação das soluções existentes para os Sistemas de Hiperdocumentos Distribuídos, em termos de soluções preventivas e corretivas, podendo estas ser de funcionamento manual, semi-automático e automático. Também é objeto desta a elaboração de uma solução (ou minimização) para o problema da integridade das informações disponibilizadas em Sistemas de Hiperdocumentos Distribuídos, tendo como enfoque principal o tratamento da inconsistência de ligações no modelo WWW, através da prototipação de uma ferramenta preventiva semi-automática.

Considera-se como solução: levantamento, leitura e análise do material bibliográfico, definição conceitual, classificação, especificação e implementação de um protótipo. A prototipação tem por objetivo mostrar a viabilidade de uma ou mais das referidas soluções para o armazenamento e recuperação de hiperdocumentos consistentes, em nível do servidor WWW do Instituto de Informática da UFRGS.

Este documento está estruturado da seguinte forma:

No capítulo 2, são apresentados as noções e conceitos referentes aos Sistemas de Hiperdocumentos Distribuídos, bem como seu histórico, estado da arte, arquitetura, requisitos e características específicas dos Sistemas HyperTed, Microcosm, VNS, Sun Link Service, Hyper-G e WWW.

No capítulo 3, é tratado especificamente o problema da inconsistência de ligações, com ênfase no ambiente WWW. São apresentadas a descrição do problema, conseqüências de sua ocorrência e classificação das soluções encontradas na bibliografia.

No capítulo 4, é apresentada a solução proposta e o protótipo implementado, sua definição conceitual, características de implementação e ambiente de prototipação.

No capítulo 5, são apresentadas as conclusões obtidas com o presente trabalho. São discutidos problemas diagnosticados no decorrer deste e que não foram contemplados em função do estabelecimento de prioridades. As soluções dos problemas detectados são sugeridas como trabalhos futuros.

São apresentados, também, a bibliografia consultada e anexos contendo códigos-fonte dos programas implementados, formulários desenvolvidos em HTML e alterações efetuadas em arquivos de configuração do servidor HTTP.

2 Sistemas de Hiperdocumentos Distribuídos

Para se definir “Sistemas de Hiperdocumentos Distribuídos” será primeiramente apresentado um histórico dos objetos hiperdocumentos. Em seguida, serão descritos noções e conceitos inerentes aos hiperdocumentos, para finalmente mostrar o estado da arte dos Sistemas de Hiperdocumentos Distribuídos, através de uma análise baseada em critérios de funcionalidade.

2.1 Histórico

Para um melhor entendimento divide-se a história dos Sistemas de Hiperdocumentos em quatro gerações:

- Primeira: Geração das idéias;
- Segunda: Implementações meramente textuais;
- Terceira: Implementação de sistemas Hipermídia;
- Quarta: Implementação de Sistemas de Hiperdocumentos Distribuídos em redes de alcance global.

Se o termo hiperdocumento for utilizado para se referir àqueles documentos onde, em seu conteúdo, são feitas referências a outros, veremos que a primeira geração remonta da época de Aristóteles (384 - 322 A.C.). Filósofo Grego, em seus escritos, utilizava pesadas referências a artigos seus e de outros autores. Ao escrever, criava uma rede de referências tal que, caso o leitor desejasse seguí-la exaustivamente, acabaria perdendo-se em citações e referências, descontextualizando-se. Os artigos de Aristóteles representam, então, o que pode se chamar de a primeira estrutura de hiperdocumento.

O primeiro sistema computacional a empregar conceitos de hipertexto foi o Memex (Memory Extender), descrito por Vannevar Bush entre os anos de 1932 e 1945. O sistema Memex, entretanto, nunca chegou a ser implementado, apesar de encontrarmos sua descrição teórica no artigo “As We May Think”, publicado na revista Atlantic Monthly em 1945.

O propósito do sistema Memex era ser um mecanismo automático, flexível e rápido onde o usuário armazenaria todas suas informações na forma de microfichas (tecnologia disponível na época) para posterior acesso. Estas seriam consultadas em projetores espalhados pela mesa, um conceito bastante similar ao hoje utilizado em ambientes orientados a janelas. O Memex seria, então, um “suplemento privado” para a memória individual.

Segundo o próprio Bush, o principal aspecto do sistema Memex seria o conceito de indexação associativa, por meio do qual um tópico levaria a outro, automática e imediatamente.

Depois da publicação, em 1945, por Bush, do sistema Memex, nada se criou em hiperdocumentos durante aproximadamente 20 anos. Neste período os cientistas quase não acreditavam que houvessem aplicações para computadores além do processamento numérico. Pode-se dizer que a primeira geração dos hiperdocumentos - definição teórica - estende-se até aproximadamente 1960 e encerra sem sistemas implementados.

A segunda fase, dos Sistemas de Hiperdocumentos Textuais, tem como seu marco referencial o sistema de Doug Engelbart [NIE 90]. Este, definido em 1962, tinha por objetivo aumentar a produtividade de escritórios, através de ferramentas computacionais de automação de escritórios e processadores de texto. Uma das partes do projeto era o NLS (oN-Line Systems) que possuía várias características de um Sistema de Hiperdocumento.

O sistema NLS foi demonstrado em 1968, durante uma sessão especial da Fall Joint Computer Conference. Doug utilizou hardware na época considerado sofisticado (mouse e projetores especiais de vídeo) para a demonstração do que viria a ser a primeira exibição de computação interativa. Apesar dos avanços apresentados, o governo americano retirou-se do projeto em 1975 quando Doug tinha criado aproximadamente metade dos conceitos de interatividade existentes até hoje.

A palavra hipertexto foi formalmente definida em 1965 por Theodor H. Nelson, criador do Sistema Xanadu. O sistema Xanadu, sem dúvida o mais ambicioso sistema de hiperdocumento idealizado até hoje, deveria ser um repositório para toda a produção literária mundial, transformando-o em um verdadeiro hiperdocumento Universal. Para Nelson a principal característica de um hiperdocumento seria a grande capacidade de armazenamento. Contrastando com esta idéia, posteriormente, Robert Glushko, em 1989, caracterizou como a principal vantagem dos hiperdocumentos a possibilidade de associação de informações.

O objetivo de Nelson era centralizar no sistema, sob a forma de hiperdocumentos, toda a literatura disponível mundialmente. As informações seriam armazenadas em uma combinação de banco de dados local e distribuído, tornando o sistema capaz de responder rapidamente a consultas utilizando informações armazenadas tanto em um computador central quanto em computadores locais.

As principais características do sistema Xanadu seriam: a possibilidade de localizar palavras, ou parte destas, em qualquer documento armazenado e o fato de um documento nunca ser apagado, o que possibilitaria a existência de várias versões de cada um dos documentos. Estas necessitariam, portanto, que os documentos fossem totalmente públicos, alterando os conceitos de “royalties” e “copyright”.

Vários outros Sistemas de Hiperdocumentos foram criados na Universidade de Brown nos anos 60, sob a chefia de Andries Van Dam.

Em 1967 o Hypertext Editing System foi o primeiro sistema de hiperdocumento que realmente foi implementado. Sua plataforma de trabalho resumia-se a um mainframe IBM/360 e 128Kb de memória RAM. Este sistema foi utilizado para produzir a documentação das missões espaciais Apollo.

O segundo Sistema de Hiperdocumentos criado chama-se FRESS (File Retrieval and Editing System) e foi concluído em 1968.

Estes dois sistemas implementaram a funcionalidade exigida (ligações e desvios) para serem considerados Sistemas de Hiperdocumentos. Considera-se como limitação a interface orientada a caracter, que obrigava o usuário a fazer uma especificação indireta dos desvios requeridos.

Os Sistemas de Hiperdocumentos desenvolvidos nas décadas de 70 e 80 constituem a terceira geração. Os hiperdocumentos passam a contar com recursos extra-texto (multimídia) e têm sua aplicação real descoberta pelo grande público. A integração da multimídia ao hiperdocumento gera a hipermídia, tônica presente atualmente na maioria dos Sistemas de Hiperdocumentos.

Considera-se como precursor desta gerações o "Aspen Movie Map" [NIE 90], desenvolvido em 1978 pelo grupo Architecture Machine do MIT. Esta aplicação foi criada para permitir uma "viagem simulada" pela cidade de Aspen na tela do computador.

O hiperdocumento foi implementado através de um conjunto de "vídeo-discos" contendo fotografias de todas as ruas da cidade. O aspecto hipermídia deste sistema provem do método como estas fotografias são recuperadas, não através de uma consulta tradicional a um banco de dados, mas como um conjunto interligado de informações.

Seguindo na utilização da Hipermídia, ao sistema "Aspen Movie Map", pode-se classificar como de terceira geração os sistemas apresentados na tabela 2.1.

TABELA 2.1 - Sistemas de Hiperdocumentos de Terceira Geração - Hipermídia

Ano	**Sistema**	**Características**
1983	KMS - Knowledge Management System	Descendente do sistema ZOG (1972-Carnegie Mellon). Criado para estações UNIX, tem como principal característica o gerenciamento eficaz de grandes hiperdocumentos.
1983	HyperTies	Desenvolvido primordialmente por Ben Shneidermann na Universidade de Maryland, desde 1987 é um produto comercial para a plataforma PC.
1985	NoteCards	Desenvolvido pelo Xerox PARC, sendo atualmente um produto comercial. Apresenta a característica de cada nodo ser um "card", onde as ligações são conexões tipadas.
1985	Intermedia	Desenvolvido pela Universidade de Brown, tem como aspecto limitante somente rodar sobre a plataforma Apple. Apresenta ligações bidirecionais.
1986	Guide	Foi o primeiro Sistema de Hiperdocumentos popular, desenvolvido para a plataforma Macintosh. É baseado em janelas de texto com "scroll" e suporta ligações a objetos gráficos.
1987	HiperCard	Bastante popular por ter sido incluído no pacote de programas gratuitos quando da compra de um computador Macintosh. Inclui uma linguagem de programação específica chamada HyperTalk e, assim como o Guide e o NoteCards, é baseado na metáfora de "cards" (cartões).

Os Sistemas de Hiperdocumentos considerados de quarta geração se caracterizam pela distribuição em computadores interligados através de redes. Este fato é de vital importância por quebrar o enfoque centralizado dos objetos-hiperdocumentos e reforçar a tendência de globalização.

Apesar de somente implementada através de sistemas desenvolvidos durante a quarta geração, a idéia da distribuição remonta ao Sistema Xanadu [NIE 90]. Theodor Nelson, ao definir o termo hipertexto, já previu nos hiperdocumentos a característica da distribuição da informação.

Os demais Sistemas de Hiperdocumentos Distribuídos serão analisados posteriormente neste capítulo.

2.2 Noções e Conceitos

Segundo Jakob Nielsen [NIE 90] a forma mais simples de se definir hipertexto é através de sua contraposição com o conceito tradicional de documento.

Nos documentos tradicionais as informações são apresentadas sob forma seqüencial, criando, portanto, uma estrutura de acesso linear. Entende-se por estrutura de acesso linear o processo de consulta ao documento seguindo apenas um contexto, definido explicitamente pelo autor, sob pena de perda de contextualização caso a consulta seja efetuada de forma diferente da originalmente definida.

Em um hipertexto, a informação deixa de ser acessada de forma unicamente seqüencial. Um mesmo hipertexto pode ser contextualizado de diferentes formas, dependendo dos caminhos que o leitor percorrer. De acordo com as relações estabelecidas entre os documentos que o compõe, pode-se adquirir diferentes contextualizações, cada uma adequada à realidade de um leitor.

Na figura 2.1 é ilustrado o acesso navegacional que o leitor realizou para poder contextualizar o conteúdo de uma maneira caso comece a ler o documento pelo tópico A, seguir a leitura pelo tópico B e após seguir para o tópico C ou outra se começar pelo tópico A passando em seguida ao tópico D e aí encerrando a leitura. O contexto criado, como pode-se notar, é diferente.

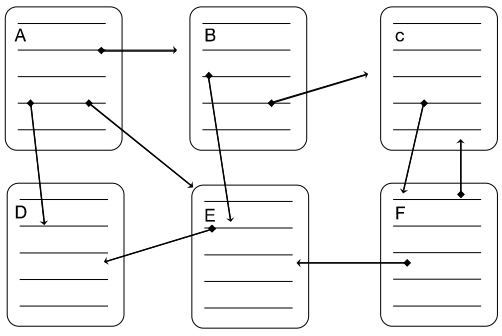

FIGURA 2.1 - Estrutura de hipertexto.

Segundo John Smith [SMI 88], hipertexto é uma aproximação da informação dirigida em que os dados são armazenados em uma rede de nodos conectados por links. Nodos são capazes de conter textos, gráficos, áudio e vídeo, bem como o código-fonte de outras formas de dados.

Um hipertexto consiste, portanto, no acesso não seqüencial ao texto de um documento através de ligações de porções de texto. A contextualização da informação é dada pelos caminhamentos efetuados através dos nodos.

Define-se hiperdocumento como sendo um objeto complexo que apresenta conteúdo, estruturas lógica, de apresentação, de navegação e que pode ser delimitado por fronteiras definidas através de referências internas ou externas.

Um nodo, segundo Vânia Fichman [FIC 91], é a unidade de informação utilizada em sistemas de hiperdocumentos cujo conteúdo geralmente pode ser exibido em uma única tela de vídeo e que possui um nome ou título que o identifica. Os nodos, idealmente, devem ser preenchidos com um único conceito ou idéia, constituindo assim unidades discretas do ponto de vista semântico e sintático.

A dificuldade em se escrever hipertextos surge no momento de se definir os nodos, uma vez que cada nodo possui uma fronteira rígida, em oposição a um texto comum, onde cada um dos parágrafos se integra com seus vizinhos. Torna-se complicado definir o que cada nodo deve conter, qual o seu tamanho e conteúdo. Esta dificuldade aumenta no momento em que se tem cada nodo podendo conter não somente texto, mas também sons, gráficos e vídeos.

Segundo Vânia Fichman, a modularização dos nodos deve ser efetuada no sentido de permitir que “uma idéia seja referenciada em qualquer local do hipertexto e que sucessores alternativos de um nodo possam ser oferecidos ao usuário”. A fronteira entre modularidade e quebra total de contexto é bastante tênue, cabendo ao autor o bom senso na modularização de seus textos.

As ligações entre nodos são definidas pelo autor no momento da criação do hipertexto. No entanto, este não tem meios de prever se o leitor utilizará ou não estas, o que torna a tarefa de autoria altamente complexa. Cada nodo, então, deve conter informação suficientemente completa para representar uma idéia e ser, também, suficientemente capaz de motivar o leitor à investigação de outras alternativas, utilizando outros caminhamentos.

Uma ligação constitui o meio de conexão física entre os nodos. São as ligações que provêem a organização da informação em hiperdocumentos, definindo o relacionamento entre dois nodos existentes na base de dados e conferindo-lhes caráter não-linear.

São funções das ligações, como diz Vânia Fichman: ”associar dois nodos cujas informações sucedem logicamente; associar uma referência feita em um documento à informação referenciada; conectar uma informação ou comentário ao texto ao qual se refere; clarear o conteúdo de mapas, gráficos ou tabelas, conectando-os a descrições e explicações e conectar uma entrada em um índice ao assunto relacionado”.

Uma ligação pode ser uni ou bidirecional, conforme o fato de existir ou não o caminho inverso. Uma ligação unidirecional é aquela em que só se pode acessar informações em um sentido, enquanto que uma ligação bidirecional é aquela em que se pode efetuar o caminhamento em ambos os sentidos.

À referência de uma ligação dá-se o nome de âncora. Uma âncora é uma área de tela sensível ao "click" do mouse, ou palavra referenciada com caracteres especiais que, quando acionada, dá origem a um caminhamento, chegando a um nodo destino. Ao se criar uma âncora, no momento da autoria do hiperdocumento, efetiva-se a ligação entre dois nodos.

Danny Lange [LAN 92], por sua vez, define um Sistema de Hiperdocumentos como um conjunto de três camadas:

- Interface com o usuário: A função desta camada é prover acesso às instâncias dos modelos e materializá-los em um periférico disponível (vídeo, impressora, etc.);
- Modelo de dados: O principal objetivo desta camada é especificar a semântica das entidades, indicando como serão compostos os hiperdocumentos. Dependendo do modelo, estas entidades podem ser nodos ou ligações de vários tipos;
- Camada de armazenamento: Provê o armazenamento das entidades definidas no modelo de dados. Sua função é suportar o modelo de dados com persistência e compartilhamento. Também é responsável pelos aspectos de distribuição dos documentos em várias camadas e repositórios.

Com a evolução da área da Computação surgem as redes de computadores, a arquitetura cliente-servidor e os ambientes distribuídos. Desta forma, baseando-se na expansão destes ambientes, surge uma nova classe de Sistemas de Hiperdocumentos: Sistemas de Hiperdocumentos Distribuídos.

A arquitetura de um Sistema de Hiperdocumentos Distribuído pode ser definida como sendo Cliente-Servidor [KAP 91]. Uma arquitetura Cliente-Servidor é aquela em que um software de execução concorrente dispõe de um grande número de informações, disponibilizadas por um processo servidor, que são acessadas simultaneamente por um grande número de usuários-final, através de processos-cliente.

A distribuição é desejável, nos Sistemas de Hiperdocumentos, pelos seguintes aspectos:

- Aumento de Desempenho: O aumento de desempenho na recuperação dos objetos dá-se através de réplicas ou partes de hiperdocumentos distribuídas em vários servidores, de acordo com a demanda local;
- Aumento da Disponibilidade: A disponibilidade é privilegiada devido ao fato de que problemas em servidores específicos não repercutirão em toda a hiperbase.

Os Sistemas de Hiperdocumentos Distribuídos possuem características extras que suportam a distribuição de nodos e ligações sobre um ambiente descentralizado. São essas [BRA 97]:

- Distribuição: A capacidade de uma máquina pode ser um fator limitante para a completitude do hiperdocumento, pois este pode ser muito grande para ser armazenado em uma única máquina.
- Cooperação: Um hiperdocumento pode ser cooperativo, ou seja, escrito por uma série de autores, em diferentes máquinas, que dominam diferentes aspectos da informação a ser contemplada. Além disso, o fato de ser distribuído faz com que um maior número de informações possam ser acrescidas ao hiperdocumento.

- Racionalização: Um hiperdocumento pode ser constituído de partes que se encontram em diversas máquinas, podendo estas ser acessadas localmente, reduzindo o tráfego nos canais de comunicação.

Segundo Gary Hill [HIL 94], existe um consenso em torno das funcionalidades que os Sistemas de Hiperdocumentos Distribuídos devem apresentar:

- A não imposição de marcas ou "tags" sobre os dados referenciados;
- A capacidade de interligar ferramentas já existentes no ambiente de desenvolvimento sem a necessidade de que estas estejam presentes quando da manipulação dos objetos gerados;
- A distribuição de dados e processos em uma rede de computadores, independente de plataformas de hardware e sistemas operacionais, caracterizando interoperabilidade;
- A não distinção artificial entre autores e leitores, o que significa que leitores podem criar ligações;
- A inclusão de novas funcionalidades de forma facilitada.

Segundo [MAI 93], a implementação de Hiperdocumentos Distribuídos deve levar em consideração aspectos de:

- Concorrência: O controle de concorrência envolve a serialização de eventos para evitar conflitos entre eventos que possam acontecer simultaneamente e com isso afetar a consistência dos dados;
- Segurança: Refere-se ao gerenciamento de acesso de múltiplos usuários (válidos ou não) a documentos privados ou públicos;
- Transparência ou Independência de Localização: Refere-se ao fato de os usuários não perceberem se o acesso aos dados está sendo realizado local ou remotamente;
- Disponibilidade: Técnicas, normalmente baseadas em replicação, que permitem ao sistema funcionar mesmo com problemas em partes de um conjunto de documentos ou máquinas.

2.3 Análise de Sistemas de Hiperdocumentos Distribuídos

Nesta seção serão apresentados sistemas que podem ser considerados como exemplos de Sistemas de Hiperdocumentos Distribuídos devido a suas características e funcionalidades.

2.3.1 HyperTed [HYP 9?]

A estrutura conceitual do HyperTED é considerada uma rede semântica implementada usando ligações bidirecionais entre quaisquer objetos de arquivo Macintosh (texto/gráfico/som), sendo que, ao menos um arquivo de cada par deve ser um arquivo de texto. A âncora, neste arquivo de texto, pode ser qualquer porção de texto (palavra, frase, sentença ou parágrafo).

Suas principais características são:

- Extensibilidade;
- Suporte a Multimídia;
- Abstração de endereçamento;
- Consistência de ligações;
- Controle de versões.

Sua principal desvantagem é que, apesar da previsão de portabilidade entre plataformas, na prática, isto nem sempre é possível.

Em relação às funcionalidades apresentadas anteriormente pode-se afirmar que:

TABELA 2.2 - Relação das Funcionalidades do HyperTED

Funcionalidade	**Resultado**
Distribuição	Apresenta (através de URL)
Cooperação	Informação Não Disponível
Racionalização	Apresenta somente em Macintosh
Não imposição de marcas	Apresenta plenamente
Interligação a ferramentas	Apresenta somente para Macintosh
Distinção entre autor-leitor	Informação Não Disponível
Extensibilidade	Apresenta somente para Macintosh
Controle de Concorrência	Apresenta
Controle de Segurança	Informação Não Disponível
Independência de Localização	Apresenta
Disponibilidade	Informação Não Disponível

2.3.2 Microcosm [MUL 97]

Microcosm é um Sistema Hipermídia Distribuído criado para o gerenciamento e a disseminação de informações digitais não-estruturadas. Foi desenvolvido como um projeto de pesquisa acadêmico na Universidade de Southamptom-Inglaterra, a partir de 1989, sendo atualmente um produto comercial disponível para a plataforma Microsoft Windows.

Sua origem é contemporânea ao Sistema WWW, adotando uma filosofia de hipermídia aberta, permite que documentos produzidos por diferentes aplicações sejam ligados mesmo mantendo o seu formato original.

O Microcosm tem sido utilizado em um grande número de projetos, incluindo a entrega de material educacional, arquivos multimídia, manutenção de documentos “on-line” e gerenciamento de informações corporativas [DER 95].

A consulta aos documentos é bastante abrangente, permitindo, entre outros, que uma âncora leve a vários documentos destino e que o usuário faça consultas através de questões.

Os múltiplos tipos de ligações são armazenadas em um ou mais bancos de dados de ligações ou "linkbases". A abstração da separação foi utilizada em todo o projeto do Microcosm. Há separação entre a exibição de documentos (através de exibidores) e o processamento das informações, realizado através de filtros. Exibidores e filtros são processos separados que interagem usando um sistema de intercâmbio de mensagens.

Exibidores são programas que possibilitam ao usuário a exibição de uma classe de documentos. Os exibidores incluídos no Microcosm tratam as seguintes classes de documentos:

- Texto: ASCII e RTF
- Gráficos: BMP, JPEG e PNG
- Vídeo/áudio: AVI e WAV
- Animações Microcosm: Seqüência de documentos gráficos
- Microcosm Mimic: Uma forma de roteiro guiado.

Além dos formatos citados acima, qualquer programa que manipule formatos de dados pode ser conectado ao Microcosm. Processadores de texto, navegadores WWW, banco de dados e planilhas eletrônicas podem atuar como exibidores.

Filtros são processos que provêem funcionalidades aos exibidores e aos usuários. Eles recebem mensagens, executam ações particulares e normalmente geram como respostas mensagens que são enviadas a exibidores e a outros filtros.

O modelo Microcosm disponibiliza múltiplos tipos de ligação:

- Ligações Específicas: A âncora é uma seleção particular em um documento e o destino também é uma seleção particular em outro ou no próprio documento;
- Ligações Locais: A âncora é qualquer ocorrência de uma seleção particular de texto em um documento, sendo que todas as ocorrências desta seleção podem levar ao documento destino. O destino deve ser uma seleção em outro ou no próprio documento;
- Ligações Genéricas: A âncora é qualquer ocorrência de uma seleção particular de texto em qualquer documento e todas as ocorrências desta seleção podem levar ao documento destino. As Ligações Genéricas oferecem grandes vantagens em nível de produtividade pois, pelo custo de uma Ligação Específica, várias ligações podem ser disponibilizadas. Além disso qualquer novo documento introduzido na hiperbase tem acesso imediato a todas as Ligações Genéricas previamente definidas;
- Ligações Dinâmicas: Permite que se criem ligações que não foram definidas explicitamente pelo autor do hiperdocumento. Pode-se citar como exemplo ligações entre partes de informação geradas automaticamente a partir de análises estatísticas do conteúdo do Banco de Ligações.

A abstração da separação de ligações apresenta as seguintes vantagens:

- A utilização de um modelo que é extensível. O modelo funciona através da escolha de um objeto e da definição de uma ação sobre ele, por exemplo, “execute esta ligação”;
- A navegação é realizada através das ligações como um processo distinto. A base de ligações padrão do Microcosm utiliza um simples banco de dados que, entretanto, pode ser substituído por um algoritmo dinâmico com busca textual;
- A utilização de um rico conjunto de tipos de ligações, não restringindo-se a ligações do tipo ponto-a-ponto. Informações contextuais são armazenadas na âncora de cada ligação, sendo que estas podem ser consultadas quando a ligação é resolvida. O conjunto de tipos de ligações existente contém desde a localização específica do documento até uma frase ou palavra de um contexto;
- As ligações podem ser aplicadas a qualquer meio pois são armazenadas separadamente;
- Diferentes conjuntos de ligações podem ser aplicadas a um mesmo documento, através da simples troca da base de ligações. Isto permite que diversos usuários tenham diferentes perspectivas de um mesmo material;
- Um Sistema de Gerenciamento de Documentos provê uma abstração de endereçamentos entre documentos e suas localizações. Este sistema de Gerenciamento pode, também, armazenar outras informações como descrições textuais, nome do autor e palavras-chave.

A arquitetura do sistema Microcosm pode ser vista na figura 2.2.

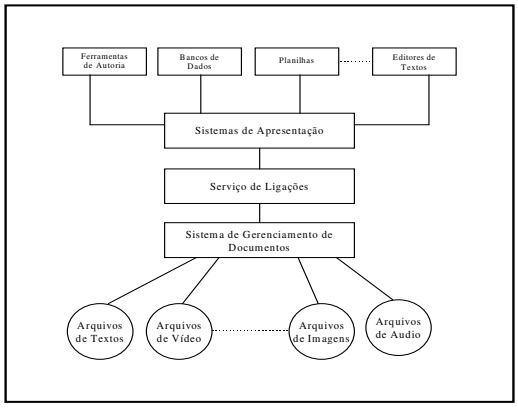

FIGURA 2.2 - Arquitetura do Microcosm.

Em relação as funcionalidades apresentadas anteriormente pode-se afirmar que:

TABELA 2.3 - Relação das Funcionalidades do Microcosm

Funcionalidade	**Resultado**
Distribuição	Apresenta
Cooperação	Informação Não Disponível

Racionalização	Apresenta somente em MS-Windows
Não imposição de marcas	Apresenta
Interligação a ferramentas	Apresenta plenamente
Funcionalidade	**Resultado**
Distinção entre autor-leitor	Apresenta plenamente
Extensibilidade	Apresenta (via Separação de Ligações)
Controle de Concorrência	Informação Não Disponível
Controle de Segurança	Informação Não Disponível
Independência de Localização	Apresenta
Disponibilidade	Apresenta (via Banco de Dados)

2.3.3 VNS (Virtual Notebook System) [HIL 94]

O VNS (Virtual Notebook System) é um Sistema de hiperdocumentos Distribuído projetado para permitir a aquisição e compartilhamento de informações científicas.

Sua principal utilização dá-se por grandes grupos de pesquisa em universidades, sendo que, neste caso, o acesso distribuído às informações disponíveis é essencial. O VNS é baseado no banco de dados relacional SyBase e foi desenvolvido especialmente para a plataforma X-Windows.

O modelo utilizado na construção do VNS é baseado em grupos de trabalho, onde cada grupo possui um servidor de trabalho em grupo, sendo que atua também como interface entre vários grupos de trabalho ou serviços externos de informações. [HIL 94]

Todos os dados no VNS são armazenados em um banco de dados relacional. Quando uma página do notebook está para ser exibida, o programa recupera toda a informação sobre objetos e ligações do banco de dados e cria a janela dinamicamente. Caso a página referencie dados que não estão armazenados no servidor local, o servidor "*gatekeeper*" resolve as referências através de mapeamento recurso/localização, e posteriormente contata o servidor necessário.

Em relação às funcionalidades apresentadas anteriormente pode-se afirmar que:

TABELA 2.4 - Relação das Funcionalidades do VNS

Funcionalidade	**Resultado**
Distribuição	Apresenta
Cooperação	Apresenta plenamente
Racionalização	Apresenta
Não imposição de marcas	Informação Não Disponível
Interligação a ferramentas	Informação Não Disponível

Distinção entre autor-leitor	Informação Não Disponível
Extensibilidade	Informação Não Disponível

Funcionalidade	Resultado
Controle de Concorrência	Apresenta
Controle de Segurança	Apresenta
Independência de Localização	Apresenta plenamente
Disponibilidade	Apresenta plenamente

2.3.4 Sun Link Service [PEA 89]

O Sun Link Service é um produto distribuído pela Sun Microsystems como parte do ambiente de desenvolvimento NSE (Network Software Environment). Foi desenvolvido para criar e manter relações bidirecionais consistentes entre ferramentas de aplicações autônomas.

Ao contrário da maioria dos sistemas de hipertextos monolíticos (que armazenam os dados e as ligações conjuntamente), as ligações e os dados referenciados são armazenados separadamente e tratados por camadas independentes. As operações sobre os objetos, como edição e armazenamento, são realizadas independentemente pelas respectivas aplicações.

O Serviço de Links armazena somente uma representação destes objetos ou nodos, permitindo que estes sejam representados em qualquer formato de dados, e que qualquer aplicação seja utilizada.

A maioria das funções de operação de ligações são providas por rotinas externas, que definem o protocolo de integração com o Serviço de Links.

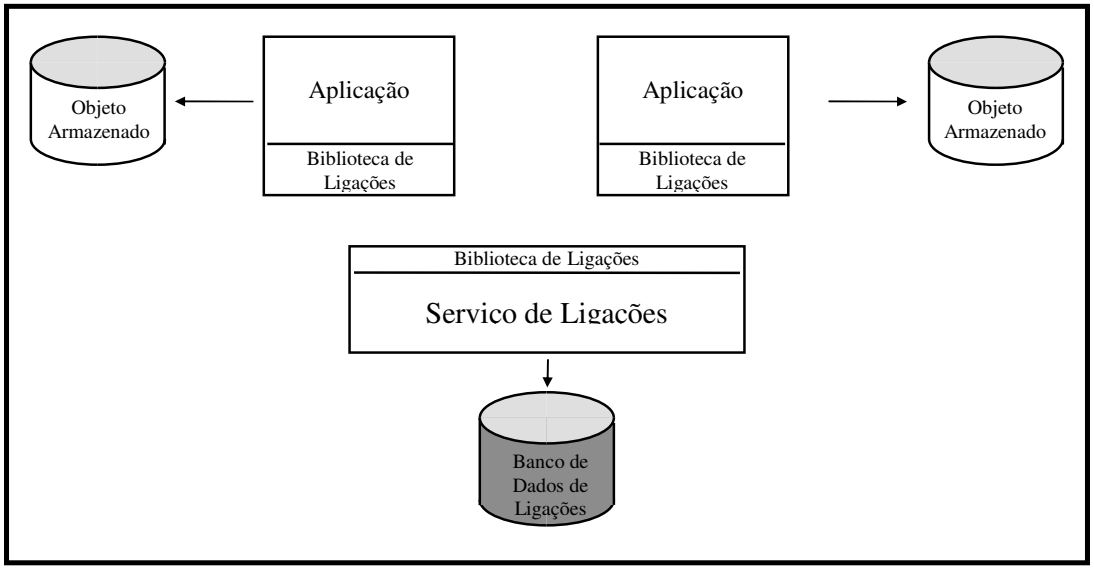

FIGURA 2.3 - Arquitetura do Sun Link Service.

Ao contrário de muitos sistemas, que necessitam que os objetos a serem ligados tenham um formato pré-estabelecido e totalmente formalizado, como um cartão ou um documento, o Serviço de Ligações permite que estes tenham quaisquer formato e tamanho. Podem ser objetos ligáveis textos, porções de texto e ponteiros, entre outros. O Serviço de Ligações requer que as aplicações geradoras referenciem os objetos que contêm ligações com uma marca específica.

Em relação as funcionalidades apresentadas anteriormente pode-se afirmar que:

TABELA 2.5 - Relação das Funcionalidades do Sun Link Service

Funcionalidade	**Resultado**
Distribuição	Apresenta
Cooperação	Informação Não Disponível
Racionalização	Apresenta
Não imposição de marcas	Apresenta imposição de marcas
Interligação a ferramentas	Apresenta
Distinção entre autor-leitor	Informação Não Disponível
Extensibilidade	Apresenta
Controle de Concorrência	Apresenta
Controle de Segurança	Apresenta
Independência de Localização	Informação Não Disponível
Disponibilidade	Apresenta

2.3.5 Hyper-G [AND 94]

Hyper-G é um Sistema de Hiperdocumentos Distribuído em desenvolvimento desde 1990 na Universidade de Tecnologia de Graz-Áustria. Este sistema baseia-se nos conceitos introduzidos pelo Xanadu e Intermedia.

Foi projetado para ser um sistema hipermídia de propósito geral, estruturado, multi-protocolo e multi-usuário, que funciona como uma aplicação Cliente-Servidor sobre a Internet e redes locais.

Hyper-G apresenta três mecanismos de navegação:

- Navegação estrutural;
- Hiperligações;
- Navegação através de pesquisas.

A utilização destes mecanismos combinados com visualizadores baseados em mapas de ligações e visões da estrutura provêm um poderoso apoio no sentido de minimizar o problema da desorientação gerado quando da navegação nos hiperdocumentos.

Os programas-cliente para o Hyper-G tanto na plataforma UNIX quanto na plataforma Microsoft Windows, implementam a total funcionalidade do modelo. Na plataforma UNIX, o programa-cliente Harmony apresenta recursos de terceira dimensão (3D), suporte a multi-línguas e exibidores integrados para texto, imagens, filmes, áudio e linguagem PostScript. Já na plataforma Microsoft Windows, o programa-cliente Amadeus tem como recurso exclusivo a capacidade de edição/autoria de hiperdocumentos.

As principais características do modelo Hyper-G são:

- Manipulação de grandes quantidades de dados multimídia;
- Disponibilização de mecanismos para a manutenção automática de hiperdocumentos com conteúdo altamente dinâmico;
- Uso eficiente e racional dos recursos computacionais e de comunicação de dados;
- Auxílio orientacional e navegacional;
- Redução na fragmentação entre servidores;
- Identificação de usuários e controle de acesso;
- Suporte a multi-linguas;
- Interoperacionalidade com sistemas existentes.

O Servidor de Ligações é um banco de dados orientado a objetos. Os objetos podem ser: descrições de documentos, ligações, âncoras, coleções, roteiros guiados e informações sobre bancos de dados remotos, bem como relações entre os objetos.

O servidor de Ligações oferece as seguintes funções:

- Designação de identificação de objetos: Cada objeto tem uma identificação única (atualmente um número inteiro de 32 bits), o que impede que dois objetos compartilhem a mesma identificação. Quando um objeto é modificado ele recebe uma nova identificação, podendo, com isso, ser distinguido da versão antiga. Caso um objeto seja removido, a identificação correspondente não poderá ser reutilizada.
- Mapeamento de identificações para os respectivos objetos: No Servidor de Ligações (e somente neste) informações complementares sobre o objeto são armazenadas. Exemplificando, no caso de documentos são armazenados: título, autor, data de criação e os dados necessários para a recuperação via servidor. Conseqüentemente, quando um documento é alterado ou trocado de servidor, somente as informações no Servidor de Ligações são atualizadas, permitindo uma consistência de ligações efetiva.
- Separação explicita de ligações dos documentos: Esta característica além de permitir uma manutenção de ligações mais aprofundada, permite que coloquemos ligações em documentos mesmo que não tenhamos direitos de escrita. As ligações, como visto anteriormente, residem no Servidor de Ligações, enquanto os documentos residem no chamado Servidor de Documentos.

- Permissão de ligações bidirecionais: Apresenta a capacidade de responder ao questionamento: "Quais outros documentos referenciam o meu documento?". Esta capacidade é importante porque quando um documento é removido ou modificado, o sistema é capaz de informar quais outros documentos são afetados, mantendo com isso a integridade das ligações. Além disso, é disponibilizado um recurso especial de navegação que possibilita ao leitor conhecer todos os documentos que tem um determinado documento como ponto de partida.

- Extensão do conceito tradicional de nodo-ligação: Através da separação ligação-documento e pela substituição das ligações organizacionais por elementos de mais alto nível, tais como: coleções, "*clusters*" e roteiros guiados. O Servidor de Ligações é atento a estas três estruturas e as utiliza para manter a consistência do banco de dados, como ações do tipo: caso um documento seja removido de um roteiro guiado, ele é automaticamente inserido no vizinho, entre outras.

- Geração de controle de acesso, que permite delegar acesso restrito a documentos individuais a certos grupos de usuários.

- Geração de estatística de utilização do sistema.

A arquitetura do Hyper-G pode ser vista na figura 2.4.

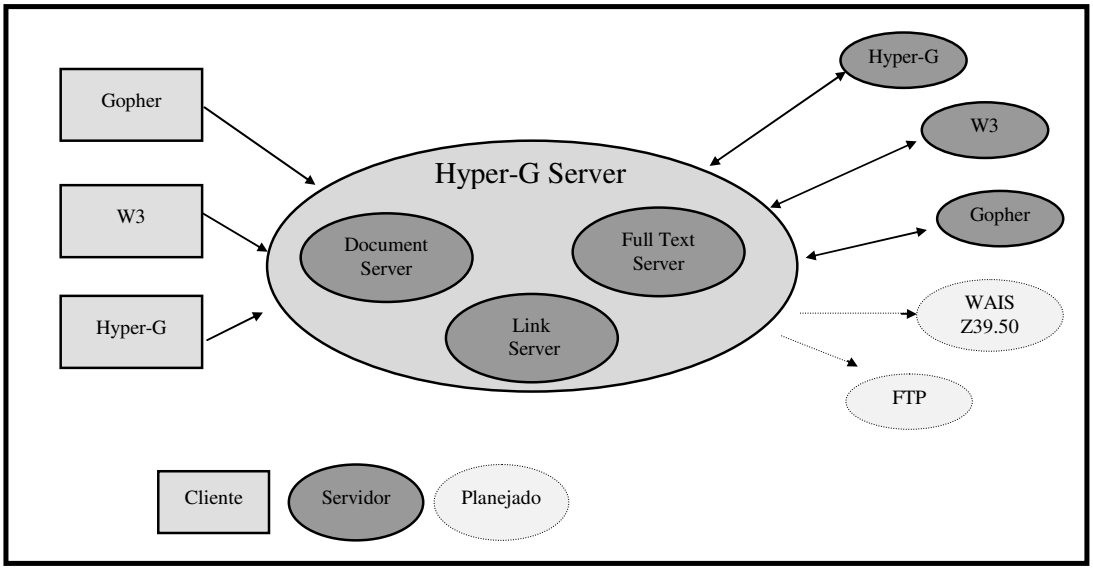

FIGURA 2.4 - Arquitetura do Hyper-G.

Em relação as funcionalidades apresentadas anteriormente pode-se afirmar que:

TABELA 2.6 - Relação das Funcionalidades do Hyper-G

Funcionalidade	**Resultado**
Distribuição	Apresenta
Cooperação	Apresenta
Racionalização	Apresenta

Funcionalidade	Resultado
Não imposição de marcas	Apresenta imposição de marcas
Interligação a ferramentas	Apresenta
Distinção entre autor-leitor	Apresenta
Extensibilidade	Apresenta
Controle de Concorrência	Apresenta
Controle de Segurança	Apresenta
Independência de Localização	Apresenta plenamente
Disponibilidade	Apresenta

2.3.6 WWW (World-Wide Web) [BER 92]

Proposto por Tim Berners-Lee, no CERN (European Particle Physics Laboratory), em Geneva, na Suíça, a partir de 1989, o WWW (World-Wide Web) é o Sistema de Hiperdocumentos Distribuídos que teve o crescimento mais acelerado nos últimos anos.

Pode-se descrevê-lo ao menos de duas formas:

- Visão organizacional: Uma iniciativa visando permitir o acesso democrático e distribuído a uma grande quantidade de documentos espalhados globalmente.
- Visão técnica: Um sistema de gerenciamento de objetos distribuídos criado para a obtenção de informações via relações textuais, através de âncoras inseridas dentro do próprio conteúdo textual.

O WWW foi desenvolvido segundo o seguinte conjunto de princípios:

- Não existência de um controle centralizado;
- Utilização de protocolos e mecanismos padronizados;
- Utilização de uma linguagem padronizada (HTML).

Pode-se considerar como vantagens do WWW [VAN 94]:

- Padrão internacionalmente reconhecido;
- Portabilidade entre plataformas ;
- Grande quantidade de Clientes-visualizadores (Mosaic, Netscape, etc.);

A arquitetura do WWW pode ser vista na figura 2.5:

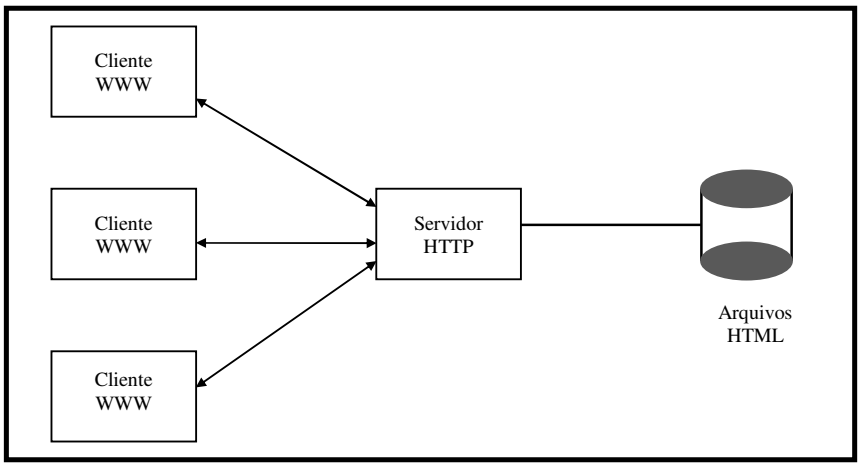

FIGURA 2.5 - Arquitetura do WWW.

Ao contrário do modelo Microcosm, que disponibiliza múltiplos tipos de ligação, o WWW apresenta somente ligações simples e, por padrão, unidirecionais.

Ligações podem ser definidas através de seleções de texto, imagens ou mapas clicáveis em um documento, tendo como destino:

- O próprio documento;
- Uma parte específica do próprio documento;
- Outro documento;
- Partes específicas de outro documento;
- Outros objetos ou recursos.

Devido a característica unidirecional das ligações, no modelo WWW, ao contrário do Hyper-G, não é possível saber quais outros documentos definem um documento específico como destino. A implementação de bidirecionalidade, caso desejada, deve ser realizada explicitamente, através da inclusão manual de ligações recíprocas, pelos autores de ambas as páginas.

Em relação as funcionalidades apresentadas anteriormente pode-se afirmar que:

TABELA 2.7 - Relação das Funcionalidades do WWW

Funcionalidade	**Resultado**
Distribuição	Apresenta plenamente
Cooperação	Apresenta
Racionalização	Apresenta plenamente
Não imposição de marcas	Apresenta imposição de marcas

Funcionalidade	Resultado
Interligação a ferramentas	Apresenta plenamente
Distinção entre autor-leitor	Apresenta
Extensibilidade	Apresenta (via CGI e outros)
Controle de Concorrência	Apresenta Inconsistência
Controle de Segurança	Apresenta Inconsistência
Independência de Localização	Apresenta
Disponibilidade	Apresenta

O modelo WWW é implementado através de programas Cliente/Servidor e de um conjunto de protocolos propostos para:

- Comunicação: HTTP (Hypertext Transfer Protocol);
- Identificação de objetos: URL (Uniform Resource Locator);
- Estruturação e visualização: HTML (Hypertext Markup Language);
- Acesso a programa externos: CGI (Common Gateway Interface).

O protocolo HTTP (Hypertext Transfer Protocol) é utilizado na gerência da distribuição dos servidores, na comunicação entre servidores e entre clientes e servidor.

A primeira parte de uma transação WWW, utilizando o HTTP, é uma requisição. O formato básico de uma requisição é:

COMMAND Location HTTP/1.0
[cabeçalhos adicionais]
[dados adicionais]

Onde a primeira linha da requisição sempre consiste de um comando seguido da localização no servidor e da versão do protocolo utilizado.

Os cabeçalhos e os dados adicionais contêm informações sobre o visualizador (browser) e dados adicionais.

A resposta do servidor usa o seguinte formato:

HTTP/1.0 xxx Status message
[cabeçalhos gerais]
[cabeçalhos de resposta]
[cabeçalhos da entidade]

[corpo da entidade]

Neste formato, *xxx* é um código de status. Os cabeçalhos e o corpo da entidade são separados por uma linha em branco.

Os códigos de estado (status) informam ao programa cliente se a transação foi válida e processada com sucesso, ou não, e qual o próximo passo.

Os códigos de status podem ser classificados em função de seu objetivo, conforme mostrado na tabela 2.8.

TABELA 2.8 - Classes dos Códigos de Status

Classe	Descrição
200	Sucesso
300	Redirecionamento
400	Erro - Cliente
500	Erro - Servidor

A tabela 2.9 lista os possíveis códigos de status e suas definições [STA 92]:

TABELA 2.9 - Códigos de Status do HTTP

Código de Status	**Definição**
200	OK - A requisição foi atendida com sucesso e a resposta desejada está sendo enviada.
201	CREATED - Este código de status segue um comando POST, indicando que um recurso ou arquivo foi criado pelo servidor e informando a nova localização do recurso.
202	ACCEPTED - Indica que a requisição foi aceita, mas o processamento ainda não foi completado.
203	PARTIAL INFORMATION - Indica que a informação retorna ainda não é definitiva.
204	NO RESPONSE - Indica que a requisição foi processada com sucesso, entretanto não é gerado conteúdo a ser retornado.
301	MOVED - Indica que o objeto requisitado encontra-se em uma nova localização, sendo que a mudança é permanente.
302	FOUND - Indica que o objeto requisitado encontra-se em uma nova localização, sendo que a mudança é temporária.
303	METHOD - Este código de status ainda não apresenta uma definição formal.
304	NOT MODIFIED - Informa ao cliente que, caso este tenha efetuada uma requisição condicional, o objeto em questão não sofreu modificações que possam atender a requisição.
400	BAD REQUEST - Indica que a requisição efetuada pelo cliente está errada ou incompleta.
401	UNAUTHORIZED - Indica que o cliente fez uma requisição a um objeto que requer uma identificação de usuário.

Código de Status	Definição
402	PAYMENT REQUIRED - Indica que o acesso a um determinado recurso requer prévia combinação de esquema de pagamento.
403	FORBIDDEN - O servidor reconhece que recebeu uma requisição mas nega-se a atendê-la, normalmente porque o servidor ou cliente não tem direito de acesso ao recurso solicitado.
404	NOT FOUND - O recurso solicitado não existe.
500	INTERNAL ERROR - O servidor reporta um erro de processamento interno e, com isso, a requisição não pôde ser tratada.
501	NOT IMPLEMENTED - Indica que o comando solicitado não está implementado no servidor.
502	SERVICE TEMPORARILY OVERLOADED - Enquanto o servidor estava acessando outro, via proxy ou gateway, este recebeu uma resposta inválida do outro servidor.
503	BUSY - Indica que o servidor encontra-se sobrecarregado.

Os cabeçalhos gerais podem ser utilizados tanto por requisições HTTP quanto por respostas e não tem ligação direta com os dados sendo transmitidos. Os dois cabeçalhos gerais são:

- Date - indica a data da transação;
- Pragma - Possibilita ao cliente enviar comandos específicos nos cabeçalhos.

Os cabeçalhos de resposta contêm informações adicionais sobre a resposta que o servidor não pôde enviar via linha de status. Por exemplo, quanto se redireciona o programa-cliente a uma nova localização, especifica-se a nova localização no cabeçalho de resposta, ao invés da linha de status:

HTTP/1.0 302 File Temporary moved
Location: http://www.inf.ufrgs.br

Os três cabeçalhos de resposta são:

- Server - Contém o nome do servidor;
- WWW-Authenticate - Informa o tipo de autenticação utilizado;
- Location - Indica a localização do recurso.

Os cabeçalhos de entidade descrevem como os dados do corpo da entidade serão retornados. Os cabeçalhos de entidade válidos e suas definições podem ser vistos na tabela abaixo:

TABELA 2.10 - Cabeçalhos de Entidade Válidos

Nome do Cabeçalho	Propósito
Allow	Informa o cliente sobre os comandos permitidos pelo servidor, como GET ou POST.
Content-Type	Indica o tipo MIME do corpo da entidade.
Content-Enconding	Possibilita que se descreva novos tipos de codificação do corpo da entidade.
Content-Lenght	Indica o tamanho do corpo da entidade em bytes.
Expires	Indica a data de expiração dos dados retornados.
Last-Modified	Indica a data e hora da última modificação do arquivo.

O corpo da entidade contém os dados retornados propriamente ditos. Todos os dados devem ser acompanhados, no mínimo, de campos Content-Type e Content-Lenght.

Os objetos ou recursos no WWW são identificados através de URL (Uniform Resource Locator). A URL é uma representação compacta da localização e do método de acesso de um recurso disponível na Internet.

O formato de um URL pode ser visto na figura abaixo:

Método	://máquina	:porta	/diretório	/arquivo
http				.html
ftp				.gif
file				.jpg
gopher		http://www.inf.ufrgs.br		.au
news		ftp://caracol.inf.ufrgs.br/teste/teste.zip		
wais		http://143.54.10.13:8080/cadastro.html		
telnet				
outros				

FIGURA 2.6 - Formato da URL

Onde:

- Método: Indica o protocolo utilizado para a transferência dos dados;
- Máquina: Indica o endereço ou número IP onde está o servidor WWW;
- Porta: Indica em qual porta, do protocolo TCP/IP, está ativo o servidor WWW;
- Diretório: Indica em qual diretório da máquina servidora está localizado o recurso desejado;
- Arquivo: Indica o nome do recurso desejado.

A linguagem HTML (Hypertext Markup Language) é usada para estruturar a informação para que esta possa ser facilmente exibida no programa cliente. Como os programas podem ser executados numa grande variedade de plataformas computacionais heterogêneas, é enfatizada a descrição do conteúdo e da estrutura ao invés da forma. Com HTML pode-se delimitar partes de documentos e utilizar como nodos.

O CGI (Common Gateway Interface) [LIU 94] é uma interface para o servidor WWW que torna possível estender sua funcionalidade. São normalmente escritos em linguagem C ou PERL e armazenados junto ao servidor WWW.

Com programas CGI é possível:

- Processar dados provenientes de formulários para transações e entrada de dados;
- Realizar pesquisas em bancos de dados;
- Criar contadores de acessos;
- Personalizar páginas WWW de acordo com as preferências dos leitores;
- Criar páginas WWW dinamicamente "*on the fly*";
- Criar sites WWW interativos.

Na prática, um programa CGI é uma forma de estender a capacidade ou integrar novas funcionalidades nesta arquitetura cliente-servidor. Uma outra forma de estender a funcionalidade de um servidor WWW seria reescrever seu código-fonte adicionando cada funcionalidade desejada. Esta solução exigeria que os programadores envolvidos conhecessem detalhes de implementação do servidor e dos protocolos utilizados na Internet, bem como do código-fonte do referido servidor.

A especificação CGI suporta dois métodos HTTP para tratar a entrada de dados em programas CGI, indicados através do campo METHOD no protocolo HTTP, GET e POST.

O método GET instrui um servidor para carregar os dados referenciados por uma URL, onde URL especifica um aplicativo CGI. A principal utilização do método GET é na realização de pesquisas ou para dirigir consultas em bancos de dados.

Por exemplo um simples aplicativo CGI que mostra as variáveis de ambiente é executado por:

<A HREF="http://www.inf.ufrgs.br/cgi-bin/test-cgi/">.

O método POST é um método no qual uma matriz de nomes e valores associados é passada para o servidor (e para um programa CGI) para análise e tratamento posteriores. O POST permite coleções de parâmetros arbitrariamente longos e complexos, ao contrário do GET, que na maioria das versões UNIX é limitado a 255 caracteres.

Os programas CGI podem, também, acessar as variáveis do ambiente, que contém informações sobre o ambiente do cliente e do servidor WWW. As variáveis do ambiente definidas para aplicações CGI provêm informações como:

- Em que endereço de rede o cliente está localizado;
- O tipo do visualizador e os tipos de dados suportados;
- O nome e a versão do servidor;
- Instruções de como receber e interpretar os dados enviados pelo visualizador.

As variáveis de ambiente podem ser classificadas como:

TABELA 2.11 - Classificação de Variáveis de Ambiente CGI

Classe	**Descrição**
Gerais	Definem as variáveis mais populares, que cada programa CGI requer para estar apto a ler entradas do servidor.
Armazenamento de Entrada	Definem as variáveis que contém os dados de entrada que estão sendo passados do servidor para o programa CGI.
Informação do Servidor	Definem as variáveis que contém as informações sobre o Servidor.
Informação do Cliente	Definem as variáveis que contém as informações sobre o Cliente (visualizador).
HTTP	São utilizadas pelos servidores para repassar, pelo servidor aos programas CGI, informações extras recebidas do cliente.

A tabela 2.12 descreve as variáveis mais comuns em ambientes UNIX.

TABELA 2.12 - Variáveis de Ambiente CGI

Variável	**Classe**	**Descrição**
GATEWAY_INTERFACE	Gerais	Descreve a versão do protocolo CGI que está sendo utilizado, normalmente CGI/1.1.
SERVER_PROTOCOL	Gerais	Descreve a versão do HTTP.
REQUEST_METHOD	Gerais	Define o método a ser utilizado para o envio de dados em programas CGI: GET ou POST.
PATH_INFO	A. Entrada	Contém informações acerca do caminho a ser utilizado pelo programa CGI.
PATH_TRANSLATED	A. Entrada	Equivalente ao PATH-INFO relativo ao sistema de arquivos.
QUERY_STRING	A. Entrada	Contém os dados de entrada caso estejamos utilizando o método GET.
CONTENT_TYPE	A. Entrada	Contém o tipo MIME que descreve como os dados estão codificados.
CONTENT_LENGTH	A. Entrada	Armazena o tamanho dos dados passados ao programa CGI.
SERVER_SOFTWARE	Servidor	Descreve o nome e a versão do servidor que está sendo utilizado.
SERVER_NAME	Servidor	Define o nome da máquina que está executando o servidor.

Variável	Classe	Descrição
SERVER_ADMIN	Servidor	Contém o e-mail do responsável pelo servidor.
SERVER_POST	Servidor	Informa a porta TCP/IP na qual o servidor está rodando.
SCRIPT_NAME	Servidor	Informa o nome do programa CGI.
DOCUMENT_ROOT	Servidor	Indica a localização do diretório root do servidor.
REMOTE_HOST	Cliente	Indica o nome da máquina-cliente que está executando o programa CGI.
REMOTE_ADDR	Cliente	Indica o endereço IP da máquina-cliente.
REMOTE_USER	Cliente	Utilizada para armazenar informações sobre acessos de determinados usuários. Normalmente está vazia.
REMOTE_GROUP	Cliente	Utilizada para autenticação de grupos de usuários.
AUTH_TYPE	Cliente	Define o esquema de autenticação que está sendo utilizado.
REMOTE_IDENT	Cliente	Utilizado para informar qual usuário, na máquina-cliente, esta utilizando o programa CGI. Não é implementada por todos os servidores.
HTTP_ACCEPT	HTTP	Contém a lista de tipos MIME que o cliente-visualizador é capaz de interpretar.
HTTP_USER_AGENT	HTTP	Armazena o nome, versão e plataforma onde o programa visualizador está sendo executado.
HTTP_REFERER	HTTP	Armazena a localização URL da página anterior que referenciou a página atual.
HTTP_ACCEPT-LANGUAGE	HTTP	Armazena a informação, passada pelo visualizador ao servidor, dizendo quais linguagens este suporta.

3 Consistência de Ligações

Visando caracterizar plenamente o problema da inconsistência de ligações, primeiramente serão apresentadas a descrição do problema e as conseqüências decorrentes. Em seguida, será visto uma classificação das soluções existentes para os diferentes ambientes de Sistemas de Hiperdocumentos Distribuídos. Finalmente são propostas soluções visando especificamente o ambiente WWW.

3.1 Descrição do Problema

Uma das mais importantes condições para a qualidade de um hiperdocumento é a manutenção da integridade e atualidade das ligações internas e externas [WHY 95]. Porém, devido as características altamente dinâmicas da Internet e de outros meios distribuídos, tornam-se problemáticas a estabilidade e a consistência dos hiperdocumentos distribuídos [TRE 95].

O problema pode ocorrer quando de alterações, nas hiperbases, nas estruturas dos documentos ou no conteúdo semântico destes. Estas alterações podem indicar tanto um problema temporário da rede ou servidor quanto um recurso que tenha sido permanentemente movido ou excluído. Acredita-se que o percentual de requisições inválidas cresça de acordo com o tempo, com a inclusão de novos recursos e com a dinamicidade intrínseca das estruturas dos Sistemas de Hiperdocumentos Distribuídos.

A falta desta consistência gera desde frustrações pessoais do usuário até perdas mercadológicas através de comércio eletrônico. Estes problemas normalmente refletem, também, uma falta de profissionalismo ou conhecimento por parte dos responsáveis pela criação, instalação e manutenção das estruturas dos hiperdocumentos [WHY 95].

Hiperdocumentos distribuídos podem ser considerados como uma grande infra-estrutura, que pode ser definida como um banco de dados de recursos de informações com sua estrutura formalmente especificada. Estas infra-estruturas contêm uma grande variedade de fontes de informação, na forma de documentos interligados e distribuídos em diferentes localizações, que sofrem manutenções por um grande número de proprietários de documentos, que, usualmente mas não necessariamente, são os autores dos mesmos.

O processo intra e extra infra-estruturas raramente é estático, tornando seu conteúdo altamente instável. Documentos podem ser movidos, alterados ou apagados, quebrando as ligações, prejudicando a sua própria infra-estrutura e as externas. Cada novo documento adicionado pode vir a resultar num crescimento exponencial nas ligações. Com o crescimento (considerado inevitável), a infra-estrutura fica complexa e de difícil manutenção, gerando o problema da integridade das ligações.

Segundo Elizabeth Gardner, que na revista WebWeek publicou uma pesquisa informal da USENET, feita entre Webmasters americanos, constata que existe uma taxa de inconsistência entre 5% a 10% para ligações externas, sem considerar aqueles documentos que alteraram conteúdos e não localizações. Em grandes mecanismos de

busca na WWW, como o Webcrawler ou Lycos, coloca-se o problema em um patamar superior de aproximadamente 30 % [GAR 96].

Atualmente, no modelo WWW, a manutenção baseia-se normalmente, em:

- Análise do arquivo de registro de erros do servidor: Estes arquivos gravam (ou tem condição de gravar) cada requisição de objetos e, se algum erro ocorre, a natureza do erro. Esta informação pode ser útil na identificação de requisições sobre objetos que foram movidos ou apagados e que portanto estão gerando requisições URL incorretas. Entretanto, normalmente apenas os administradores do servidor têm acesso a estes arquivos. A análise dos arquivos de registro, também não permite a descoberta de falhas em documentos que nunca foram acessados, nem pode auxiliar na manutenção preventiva ou em problemas com a mudança do conteúdo dos documentos.
- Análise das reclamações dos usuários: Através da análise dos avisos sobre ligações quebradas ou documentos mal-formados, o responsável pode descobrir problemas e posteriormente corrigi-los. Entretanto a manutenção não pode unicamente basear-se nas queixas dos usuários pois existem usuários que são muito tolerantes quanto a erros, se sentindo pouco a vontade de reclamar ou apontar problemas;
- Navegação periódica e manual nos documentos: Sabendo que a manutenção manual é dispendiosa, em termos de tempo e dinheiro, e além disso enfadonha, a manutenção é raramente realizada ou realizada de forma incompleta;

Face a todos estes problemas faz-se necessário que se desenvolvam novos modelos de Sistemas de Hiperdocumentos Distribuídos que tenham a solução para a inconsistência de ligações prevista no modelo e implementada na própria arquitetura ou que se crie meios automáticos para a navegação e checagem das ligações, gerando relatórios para a manutenção humana ou mesmo efetuando a manutenção automaticamente. [FIE 94].

Entretanto, a maioria dos sistemas de gerência de Hiperdocumentos Distribuídos não foi projetado prevendo a consistência de ligações simplesmente por não haver a preocupação de robusteza e para facilitar as implementações.

3.2 Classificação das Soluções

Uma contribuição desta dissertação é a classificação das soluções existentes para o problema de inconsistência de ligações em Sistemas de Hiperdocumentos Distribuídos, em um primeiro nível como preventivas e corretivas e, posteriormente como manuais, semi-automáticas e automáticas.

3.2.1 Soluções Preventivas

As soluções preventivas são aquelas que não permitem que ocorram inconsistências nem por curtos períodos de tempo. Neste caso, alterações que possam vir a gerar inconsistências não são permitidas ou são reportadas ao autor para que este tome as devidas providências.

Consistem, normalmente, da utilização de modelos de distribuição que contemplem a manutenção e consistência como característica própria ou de modificações em modelos deficientes visando a inclusão destas características. Podem ser classificadas como:

- Em nível de Ferramentas Manuais;
- Em nível de Ferramentas Semi-automáticas;
- Em nível de Ferramentas Automáticas;
- Em nível de Modelo.

3.2.1.1 Soluções Preventivas em nível de Ferramentas Manuais

As soluções preventivas manuais são aquelas em que a possibilidade de ocorrência de inconsistência é reportada ao usuário quando da execução do processo que a gerará. A ação em relação a esta possível inconsistência é de responsabilidade total do usuário.

Pode-se citar como exemplo de solução preventiva manual o sistema Sun Link Service que trata o problema da integridade de ligações de duas maneiras: implícita e explícita.

O tratamento implícito ocorre quando um usuário tenta acessar um nodo inválido. O Serviço de Ligações informa que a ligação não é mais válida e sugere a remoção definitiva da ligação.

Caso não seja possível, ou desejável, descobrir as inconsistências da forma implícita, o Serviço de Ligações provê um "explicit link garbage collection mechanism", chamado "Confirm All", que percorre todas as ligações verificando sua consistência, através de perguntas às aplicações responsáveis pela criação da ligação.

3.2.1.2 Soluções Preventivas em nível de Ferramentas Semi-Automáticas

As soluções preventivas semi-automáticas são aquelas em que a inconsistência de ligações é evitada através de processos que envolvem alguma participação, prévia ou posterior, do usuário responsável.

Como exemplo de solução preventiva semi-automática pode-se citar o protótipo da ferramenta desenvolvida como solução proposta. Neste caso, a participação do usuário é prévia, devido a necessidade de cadastramento de redirecionamentos de forma manual. A efetivação do redirecionamento, por sua vez, é realizada de forma automática através de consulta ao banco de dados implementado.

3.2.1.3 Soluções Preventivas em nível de Ferramentas Automáticas

As soluções preventivas automáticas caracterizam-se pela não necessidade de intervenção humana no processo de prevenção de inconsistências de ligações. Atuam através de processos totalmente independentes responsáveis por tarefas específicas que interagem entre si.

Pode-se considerar como exemplo de solução preventiva automática o processo realizado pelo sistema Hyper-G, que trata o aspecto de inconsistência de ligações através de um banco de dados que mantêm meta-informações sobre os documentos bem como suas relações com outros. Desde que as ligações são armazenadas neste banco de dados e não nos próprios documentos, e sendo as alterações em documentos ou em sua relações somente possíveis via servidor Hyper-G, a integridade pode ser facilmente mantida, mesmo quando a alteração atravessa a fronteira física do servidor. Se uma alteração atinge documentos armazenados em dois ou mais servidores, estes utilizam um protocolo de atualização para que a consistência esteja garantida. [KAP 95]

Outro exemplo é visto no sistema VNS. O problema da integridade das ligações é tratado como sendo interno ao banco de dados. Normalmente, Sistemas de Banco de Dados oferecem mecanismos que, operando conjuntamente com o Sistema Operacional, garantem controle de acesso, segurança dos documentos e consistência de ligações.

O Microcosm executa um tratamento ao problema da inconsistência de ligações de forma similar ao Hyper-G: mudanças nas ligações ocorrem, via Serviços da Camada de Ligações, diretamente no banco de dados de ligações e estas são espalhadas automaticamente para os outros servidores. Contudo, a garantia da integridade das ligações não pode ser considerada 100% devido ao fato de nunca ser possível afirmar que *todas* as alterações, em *todos* os servidores, foram concluídas regularmente.

O problema da inconsistência de ligações no HyperTed é tratado através de duas estratégias: Mudança no conteúdo dos documentos e mudança na localização ou nomeação dos documentos.

Desde que este sistema não utiliza marcas ou códigos de controle dentro de um documento para efetuar, sempre que uma nova ligação é criada em um documento textual, ocorre o seguinte:

- O texto selecionado é extraído e enviado a um sistema de busca em tempo-real;
- O sistema de busca compara o texto extraído com o restante do documento, incrementando o tamanho do fragmento até que ele ocorra uma única vez no documento;
- A âncora desta ligação é unicamente especificada pelo texto resultante.

Sempre que um documento é aberto e é efetuada uma requisição às ligações disponíveis, ocorre o seguinte:

- Uma busca é realizada usando o fragmento de texto previamente extraído;
- Onde ele aparecer no texto é definida uma âncora.

Usando esta abordagem é possível editar um documento como desejado. Enquanto o fragmento de texto utilizado como âncora manter-se inalterado a ligação manter-se-á íntegra. A mudança de um parágrafo que contém o fragmento de texto não altera a integridade da ligação, porque o fragmento em si mantém-se inalterado. Esta característica permite a edição do documento em qualquer editor de textos sem que haja destruição das ligações.

Para manter as ligações a documentos que mudaram de localização ou de nome, o Hyperted faz uso das características de arquivo providas pelo sistema operacional do Macintosh System 7. Este, por armazenar informações sobre o arquivo, incluindo nome, posição, tipo de arquivo, programa criador, data de criação e várias outras características, torna possível a realização de buscas rápidas por arquivos que têm suas características coincidentes com as características conhecidas do arquivo, devolvendo o arquivo mais coerente. Se ambos o conteúdo e o nome do arquivo mudarem, este sistema falha.

Pode-se, ainda, considerar como exemplos de soluções preventivas automática as soluções descritas a seguir:

3.2.1.4 Soluções Preventivas em nível de Modelo

As soluções preventivas em nível de modelo são aquelas que apresentam ou propõem alterações nos modelos utilizados para a definição dos Sistemas de Hiperdocumentos Distribuídos ou utilizam técnicas herdadas de outras áreas como garantia que não ocorrerá a inconsistência de ligações.

3.2.1.4.1. Identificador Único [TRE 95]

A manutenção da integridade das ligações pode ser auxiliada por uma revisão no conceito clássico de ligação. A simples troca no esquema de identificação de nodos pode favorecer a manutenção. Uma das formas de referência é a utilização de referências únicas, como as utilizadas na definição do formato da bibliografia de trabalhos científicos, ao contrário da utilização de localizações físicas nos servidores.

Esta forma provê uma identificação única para cada documento, de maneira altamente formalizada, o que facilita a manutenção automática. As referências tipo bibliográficas não tratam a localização do documento, somente contém seu identificador de forma não-ambígua. No ambiente WWW isto pode ser implementado através da relação URN (Uniform Resource Names) para URL (Uniform Resource Locators). Estes protocolos são parte de uma extensa estrutura onde URN é utilizado para formalizar a identificação do objeto e URL a sua localização.

Esta solução torna necessária a construção de mecanismos distribuídos para a resolução de relações URN para URL. Já existem protótipos destes mecanismos, no modelo WWW, como o localizado no Saint Joseph's College, nos Estados Unidos. Outra alternativa seria a utilização de mecanismos de "broadcast" de mensagens, como os utilizados pelo USENET News, que anunciariam publicamente relações URL e URN.

3.2.1.4.2 Banco de Dados de Ligações [HIL 95]

A separação das ligações dos documentos pode auxiliar na manutenção de grandes estruturas de hiperdocumentos. Detalhes de todos os documentos podem ser armazenados em um banco de dados de documentos, sendo que para cada documento é

designado um identificador único. Todo o processo de ligação fica baseado nestes identificadores. Portanto, se um documento é alterado ou removido, os detalhes da operação são armazenados no banco de dados, e todas as ligações mantêm-se válidas.

São exemplos de Sistemas Gerenciadores de Hiperdocumentos Distribuídos que trabalham com esta solução o Hyper-G e o Microcosm.

3.2.1.4.3 Transações Multi-Servidor [KAP 95]

Outra possível solução preventiva para o problema de inconsistência de ligações é conhecida por *Transações Multi-Servidor* ou *Servidores Colaborativos*. Quando um documento em um servidor é modificado (ou excluído) o servidor que armazena este documento procede como um *coordenador*, contatando e informando todos os outros servidores que estão envolvidos na operação. Quando todos os outros servidores têm conhecimento da operação, o coordenador informa que a alteração foi tornada permanente. Alguns detalhes a mais são necessários para garantir que a operação foi efetuada em todos os servidores participantes, mesmo em caso de queda do servidor durante a operação.

Implementada no *Xerox Distributed File System* e em uma versão primitiva do *Hyper-G*, esta solução apresenta problemas de desempenho e confiabilidade em situações específicas. Por exemplo: quando existem muitos servidores envolvidos (acima de 1000) referenciando um determinado documento. Nestes casos, todos estes servidores devem ser contatados e responder antes que a alteração torne-se permanente, gerando um grande tráfego na rede, e também requerendo que todos os servidores estejam ligados e funcionando para que a transação seja completada. Como o número de servidores participantes tende a crescer, a probabilidade de que todos estejam ligados simultaneamente diminiu, o que torna praticamente impossível alterar um documento.

3.2.1.4.4 Sistemas Mestre-Escravo [KAP 95]

Nos sistemas que utilizam a solução *Mestre-escravo*, há um servidor primário (mestre) e um número de servidores secundários (escravos). O servidor primário mantêm uma cópia principal de todas as requisições de alteração dos objetos replicados. Os escravos são atualizados pela recepção de mensagens de notificação geradas pelo servidor primário ou fazendo diretamente cópias deste. Os usuários (utilizando clientes) podem ler os dados de ambos, principal ou secundários, mas só têm permissão para escrever no servidor principal.

Esta solução é apropriada para aplicações onde os objetos são lidos com alta freqüência e com baixas taxas de alteração, como, por exemplo, o serviços de páginas amarelas da SUN (NIS).

O servidor principal central facilita a manutenção da consistência perante as requisições de alteração. Uma desvantagem dessa solução surge da necessidade de que o servidor principal esteja ligado e rodando para a execução de alterações.

3.2.1.4.5 Controle Distribuído de Atualizações [KAP 95]

A solução preventiva, disponibilizada através do *Controle Distribuído de Atualizações,* permite que qualquer servidor que mantenha uma cópia de um objeto faça as alterações neste, sem a interferência de um servidor coordenador central. Mesmo que alguns servidores estejam sem condições de utilização, a operação é realizada eliminando assim a possibilidade de conflitos de consistência.

Esta solução requer que um servidor conheça todos os outros servidores que têm cópias dos objetos. Numa situação idealizada, todas as cópias, em todos os servidores, serão idênticas, mas, devido a problemas como falhas de rede e também questões de desempenho global, pode ser possível, e até mesmo desejável, que as notificações de alteração não sejam imediatas. Esta característica leva a uma forma especial de consistência, frágil, onde todas as cópias convergem para um mesmo valor em algum intervalo de tempo, após o término das alterações.

A fim de garantir que todas as consultas sejam realizadas em cópias atualizadas, a solução adota uma técnica de consenso: Um servidor, ao receber uma requisição de um usuário sobre um determinado objeto, consulta alguns outros servidores próximos para verificar se sua cópia esta atualizada. Caso a maioria das cópias nos outros servidores seja idêntica a sua, ele assume que a cópia está correta e a disponibiliza ao usuário.

A principal vantagem dessa solução é sua robustez. Não há um único ponto de falha, levando-se em conta que uma falha, para ser considerada, deve ter atingido ao menos 51% dos servidores. Este índice inviabiliza uma alteração e compromete a coerência dos documentos.

3.2.2 Soluções Corretivas

As soluções corretivas são adotadas em Sistemas de Hiperdocumentos Distribuídos que não dispõem de manutenção de ligações de forma natural, através de seus próprios modelos e implementações. Podem, ainda, ser classificadas como [TRE 95]:

- Manuais;
- Semi-automáticas;
- Automáticas.

3.2.2.1 Soluções Corretivas Manuais

As Soluções Corretivas Manuais são as mais comuns de se encontrar em documentos pessoais ou pequenas hiperbases.

Uma solução corretiva manual é utilizada para assegurar que, quando a hierarquia de diretórios do servidor de arquivos é atualizada, as ligações correspondentes também o sejam. Utilizando por exemplo, servidores WWW baseados no sistema UNIX, isto pode ser feito através da utilização de ligações de arquivos próprias do sistema operacional.

Servidores baseados em computadores Macintosh podem lançar mão da utilização de "*aliases*".

O exemplo mais comum de solução manual é a inclusão de um documento, genérico (para todos os documentos/usuários do servidor) ou específico (para cada documento/usuário) contendo as seguintes características:

- Indica que o arquivo não mais existe ou mudou de localização;
- Exibe o novo endereço URL;
- Provê uma ligação clicável para o novo endereço.

A figura 3.1 mostra um exemplo de solução corretiva manual:

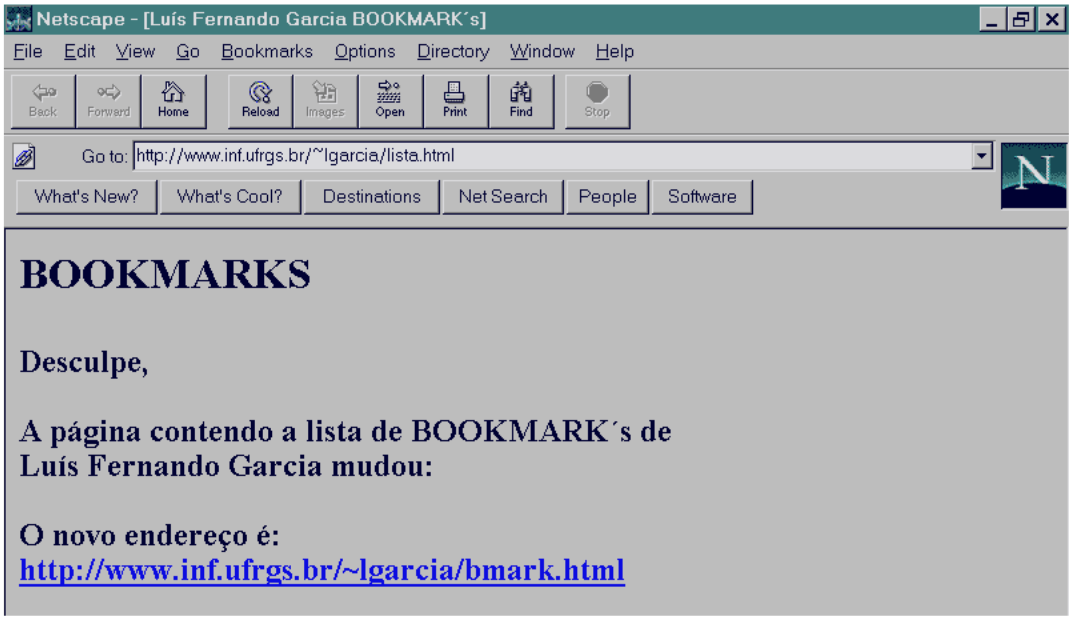

FIGURA 3.1 - Exemplo de Solução Corretiva Manual

Uma pequena vantagem apresentada pelas Soluções Corretivas Manuais decorre do fato de não serem necessárias ferramentas adicionais para a verificação da integridade, além das usuais: Editor e visualizador de documentos em HTML.

Por outro lado, as deficiências encontradas na solução manual são:

- Dificuldade em detectar as inconsistências de ligações de forma manual, pois navegar por todo o hiperdocumento exigiria grande dispêndio de tempo e esforço;
- A sobrecarga do responsável pelo servidor, pelo fato deste ter que detectar e corrigir todas as ocorrências de inconsistências;
- As ligações de arquivos, no caso de servidores baseados em UNIX, muitas vezes são válidas apenas em um determinado período de tempo. Após este período, o administrador é obrigado a alterar fisicamente o documento original;
- Os autores dos documentos que possivelmente apontam para documentos com problema não são (e talvez não possam ser) notificados para que alterem suas ligações de origem.

- Duplicação de esforços, visto que uma mesma hiperbase pode ser analisada por diferentes verificações autônomas, sendo que, deste modo não há compartilhamento de resultados e soluções.
- Geração de alto tráfego na rede devido ao fato de cada ligação de origem ter de ser analisada e percorrida.
- Utilização de visualizadores convencionais no processo de investigação, os quais exigem que todo o conteúdo do documento (ao invés de somente o cabeçalho) seja transferido para a verificação do estado das ligações, gerando com isso tráfego desnecessário.

3.2.2.2 Soluções Corretivas Semi-Automáticas

As soluções corretivas semi-automáticas atuam indicando ao responsável pela verificação da integridade da hiperbase quais ligações apresentam problemas. Normalmente, apenas é indicado o problema e apresentadas soluções padronizadas, sem que no entanto haja algum tipo de correção automática.

Este enfoque libera o responsável da tarefa de busca pelas ligações defeituosas, mantendo, entretanto, uma necessidade de intervenção manual para a correção.

O funcionamento da maioria desta ferramentas dá-se através do uso de robôs ou "spiders", que periodicamente navegam por todas as ligações de uma hiperbase assegurando-se que estas estão funcionando e são válidas. Existem exemplos de ferramentas que podem estender este conceito para hiperbases externas.

São exemplos deste tipo de ferramenta:

- MOMspider;
- URL-minder;
- html_analyser;
- CHECKER;
- Link Verifier;
- Katipo;
- ChURL;
- Checkbot;
- Astra Site Manager
- Info Link Checker
- LinkBor
- LinkScan
- SiteSweeper
- Netscape LiveWire
- WebMapper

A figura 3.2 mostra um exemplo de solução corretiva semi-automática:

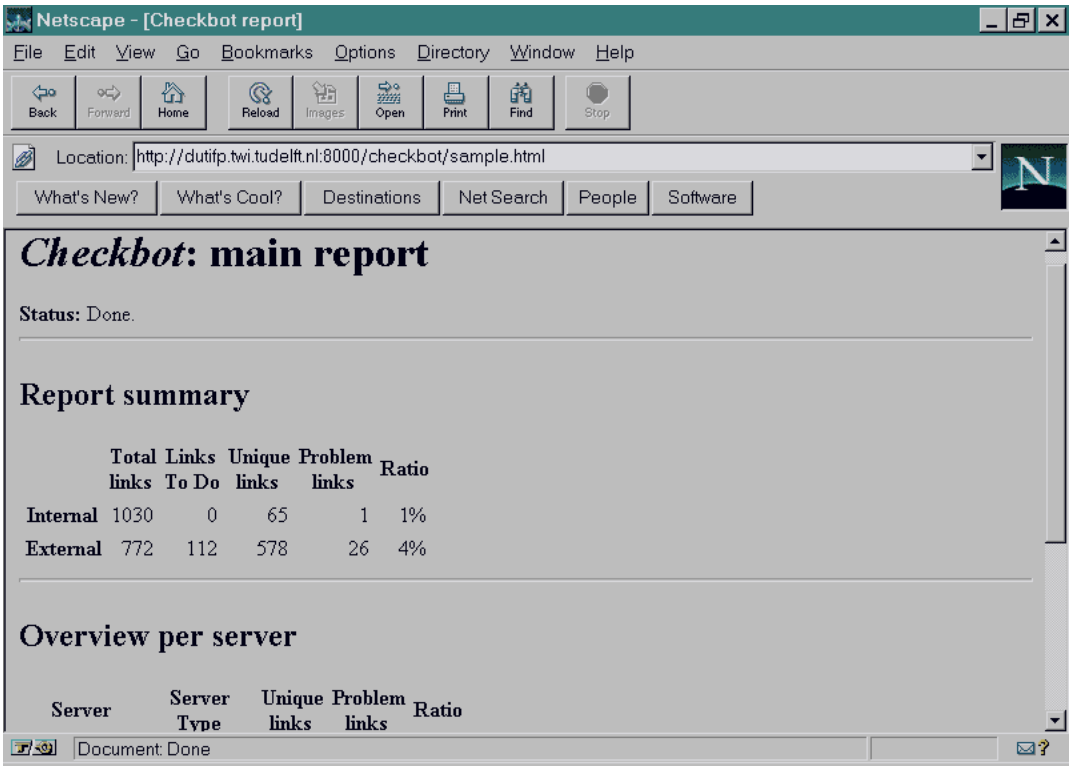

FIGURA 3.2 - Exemplo de Solução Corretiva Semi-Automática

3.2.2.3 Soluções Corretivas Automáticas

As soluções corretivas automáticas para o modelo WWW ainda não foram totalmente desenvolvidas devido a uma série de características técnicas do modelo. Entre estas a principal a ser ressaltada é o fato das ligações estarem embutidas no corpo do documento, dificultando a análise e posterior manutenção automática. Outro empecilho é a necessidade de se realizar uma análise semântica apurada do documento e, possivelmente, a utilização de técnicas de inteligência artificial para que a ferramenta possa verificar se uma ligação específica está com problemas, se estes são temporários ou permanentes, decidindo em que casos a correção deve ser efetuada e o responsável notificado.

Mesmo apresentando estas dificuldades operacionais, o processo automático tem características que o tornariam mais efetivo. Um programa de teste automático, utilizado no ambiente WWW, por exemplo, não precisaria transferir totalmente o documento a ser verificado com a requisição do protocolo HTTP GET, mas sim, com a requisição HEAD, transferir apenas as meta-informações existentes nos cabeçalhos, economizando recursos da rede.

Com o uso da manutenção automática pode-se evitar, também, o problema da duplicação de esforços apresentado pela solução manual, já que um mesmo programa automático pode tratar de várias estruturas, de diferentes proprietários e após, distribuir e compartilhar o resultado.

3.3 Soluções Atuais para World-Wide Web

Optou-se pela classificação em separado das soluções disponíveis atualmente para o modelo WWW por ser este considerado como principal objeto de estudo e também por este ser o modelo contemplado por uma prototipação de solução.

As soluções (ou atenuações) apresentadas para o problema da inconsistência de ligações no modelo WWW encontram-se classificadas da seguinte forma:

- Soluções Corretivas Manuais
    - Soluções Corretivas Manuais em nível de Documentos
    - Soluções Corretivas Manuais em nível de Servidores
- Soluções Corretivas Semi-Automáticas
    - Soluções Corretivas Semi-Automáticas em nível de Ferramentas Externas.

Não foram encontradas na bibliografia soluções preventivas, mesmo manuais, semi-automáticas ou automáticas, assim como soluções corretivas automáticas que contemplem o modelo WWW. O protótipo desenvolvido como exemplo de solução neste trabalho, classificado como sendo preventivo semi-automático, foi considerado como uma das contribuição desta dissertação.

3.3.1 Soluções Corretivas Manuais em nível de Documentos

As soluções em nível de Documentos são aquelas que podem ser realizadas diretamente embutidas dentro dos documentos descritos em HTML, geralmente implementadas pelos próprios autores.

São exemplos de soluções em nível de Documento o Redirecionamento Manual, utilização da marca META e de comandos JavaScript.

3.3.1.1 Solução Corretiva Manual via Redirecionamento Manual

A solução implementada via Redirecionamento Manual consiste, normalmente, no uso de documentos temporários. Estes são criados pelo autor do documento que geraria uma inconsistência de ligações, são armazanados com a mesma localização absoluta do documento original e têm a função de relatar ao usuário-leitor que o documento requisitado teve sua localização alterada ou até mesmo que ele não encontra-se mais disponível para acesso.

A implementação dessa solução dá-se através de um documento codificado em HTML contendo:

- Um aviso que o documento foi alterado/apagado;
- Uma ligação clicável que remete o usuário à nova localização.

Um exemplo desta solução pode ser observado na figura 3.3:

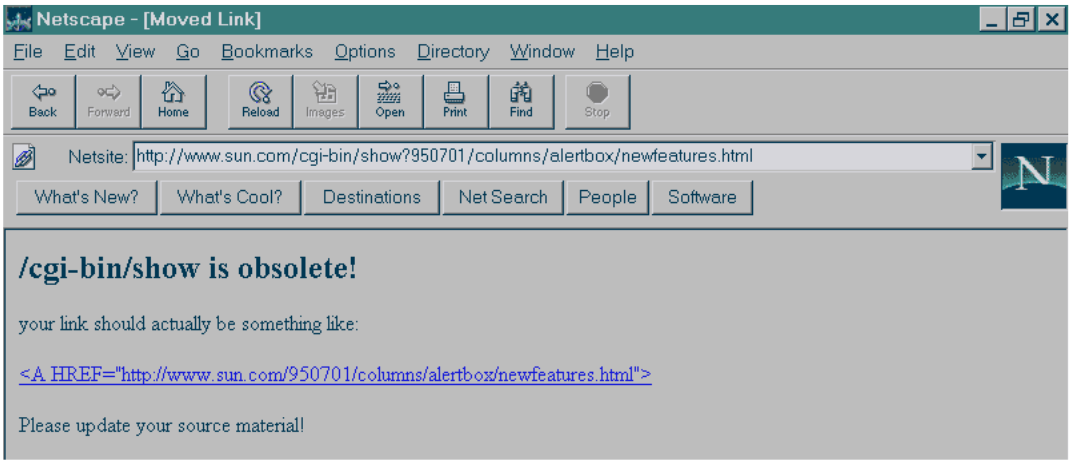

FIGURA 3.3. Exemplo de Redirecionamento Manual

3.3.1.2 Solução Corretiva Manual via Marca META [STO 95]

A solução implementada via Marca META baseia-se na característica especial do HTML 3, implementada no visualizador Netscape e posteriores.

Um elemento META pode ser utilizado para incluir pares de nome/valor, descrevendo propriedades de um documento, como autor, data de expiração e, neste caso, tempo de redirecionamento automático.

Com a marca META pode-se implementar um tipo especial de redirecionamento manual, sem intervenção do usuário. Explicitamente são criados documentos HTML exibindo a antiga e a nova localizações, implementando as mesmas opções de ligações clicáveis, mas, se dentro de um certo período de tempo, o próprio usuário não acionar a ligação que leva a localização correta, o navegador o redireciona automaticamente.

Uma desvantagem da utilização da marca META deve-se ao fato de que nem todos os visualizadores de documentos WWW a implementarem. Sabe-se que os visualizadores Netscape, da Netscape Corporation e Internet Explorer, da Microsoft, implementam-na sem problemas.

A marca META é utilizada seguindo a seguinte sintaxe:

```
<META HTTP-EQUIV="refresh" CONTENT="xx;url=http://www.nova>”
```

Onde CONTENT = xx define que o tempo de espera será de xx segundos e www.nova indica a nova localização do documento desejado.

Um exemplo da solução através da marca META pode ser observado na figura 3.4:

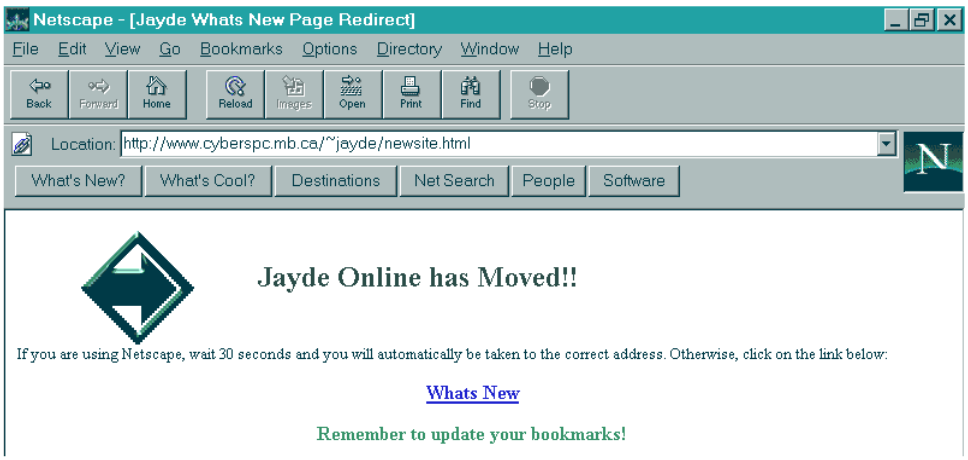

FIGURA 3.4. Exemplo de redirecionamento via Marca Meta

3.3.1.3 Solução Corretiva Manual via Comandos JavaScript [ACR 97]

A solução corretiva manual implementada através de Javascript baseia-se em um comando que redireciona a janela para uma nova localização. Este comando deve ser inserido no início de uma página HTML.

O comando Javascript necessário pode ser visto no trecho de código-fonte nostrado a seguir:

```
<HTML>
<HEAD>
<SCRIPT Language=\"JavaScript\">
window.location = http://143.54.10.13:8080
</SCRIPT>
</HEAD>
<BODY>"
</BODY>
</HTML>
```

Pode-se considerar como desvantagens da utilização de redirecionamento através de JavaScript a não possibilidade de informar ao leitor o redirecionamento e o não funcionamento em todos os programas visualizadores, especialmente aqueles mais antigos ou orientados a caracter.

3.3.2 Soluções Corretivas Manuais em nível de Servidores

As soluções em nível de Servidor são aquelas implementadas diretamente nos programas servidores HTTP. As implementações normalmente são realizadas somente pelos administradores dos servidores e dão-se através de diretivas especiais acrescentadas nos arquivos de configurações destes.

Pode-se considerar como uma vantagem das Soluções Corretivas Manuais em nível de Servidores o fato de, normalmente, as alterações nos arquivos de configuração sejam simples, sendo na maioria das vezes, a inclusão de apenas uma linha.

Como desvantagens da utilização das Soluções Corretivas Manuais em nível de Servidores temos:

- As alterações nos servidores podem ser efetuadas apenas pelo responsável, impossibilitando soluções em nível de usuário;
- Após as alterações, o servidor deve obrigatoriamente ser desligado, o que, em grandes hiperbases com grandes taxas de acesso e em especial com suporte a comércio eletrônico, torna-se uma característica altamente indesejável.

As Corretivas Manuais em nível de Servidores são normalmente implementadas através da diretiva ou regra REDIRECT, disponibilizada tanto no servidor NCSA quanto no CERN.

O servidor CERN [CER 96] permite o mapeamento entre URLs “virtuais” em nomes de arquivos físicos. Este mapeamento dá-se através de acréscimo da regra REDIRECT no arquivo de configuração */etc/httpd.conf*.

Quando documentos ou árvores completas de documentos são movidas de um servidor para outro ou há mudança interna de localização, pode-se utilizar a regra REDIRECT.

A definição formal da regra REDIRECT é:

Redirect *template result*

Onde *template* refere-se a antiga localização que, quando é solicitada, é redirecionada a URL explicitada por *result*. O formato da URL em *result* deve ser completo, isto é, contendo http:// e o nome da máquina.

Um exemplo de redirecionamento pode ser visto abaixo:

Redirect /~lgarcia/* http://www.lgarcia.com

O servidor NCSA também permite o mapeamento entre URLs virtuais e arquivos físicos, sendo que este, além de apresentar a diretiva REDIRECT, com definição formal idêntica a do servidor CERN, apresenta duas novas diretivas: RedirectTemp e RedirectPermanent. A diferença entre elas deve-se ao fato das diretivas Redirect e RedirectTemp retornarem código HTTP 302, enquanto RedirectPermanent retorna código HTTP 301.

As diretivas REDIRECT, nos servidores NCSA e APACHE, são efetuadas no arquivo de configuração SRM.CONF.

No servidor Netscape Communications Server [NET 96], desenvolvido pela empresa Netscape, a diretiva de redirecionamento é igualmente denominada Redirect, entretanto sua definição formal e localização diferem dos servidores anteriores.

O arquivo que suporta a diretiva Redirect chama-se obj.conf e sua definição formal é equivalente a:

```
NameTrans fn=redirect from=/VELHA_URL url-prefix=NOVA_URL
```

3.3.3 Soluções Corretivas Semi-Automáticas em nível de Ferramentas Externas

As soluções em nível de Ferramentas Externas são aquelas disponibilizadas através de programas externos tanto aos visualisadores quanto aos servidores.

3.3.3.1 MOMspider

MOMspider (Multi-Owner Maintenance Spider) é um robô (programa que percorre automaticamente estruturas fazendo ações diversas) especializado na manutenção de estruturas distribuídas de hiperdocumento, especialmente, em servidores WWW.

O programa foi desenvolvido em linguagem PERL e executa somente em servidores baseados em sistemas UNIX.

Sua operação é baseada em um arquivo de configuração que contém as ações específicas que ele deve tomar. As tarefas são utilizadas para descrever uma estrutura (servidor) específica que deve ser verificada durante o processo de navegação. Cada tarefa contém o tipo de navegação (servidor, árvore ou proprietário), o nome da estrutura (para posterior referência), o documento topo da hierarquia, a localização onde será armazenado o relatório de saída, o endereço eletrônico (e-mail) do responsável pela estrutura e um conjunto de parâmetros que regulam quando uma mensagem eletrônica deverá ser emitida em resposta a situações específicas.

Para cada tarefa, o MOMspider percorre a estrutura, a partir do documento topo, descendo para os nodos folha. As informações produzidas em cada tarefa são formatadas em HTML e colocadas na localização especificada no arquivo de configuração. São exemplos de saída de uma tarefa MOMspider:

- Data/hora da execução do programa e características ativadas;
- Ligação de hipertexto para uma versão anterior do relatório de saída;
- Informações do cabeçalho do documento (título, data da modificação, etc);
- Lista de referências cruzadas de nodos fonte e destino.

São procurados quatro tipos de modificações em documentos que, reveladas, auxiliarão na manutenção da estrutura:

- Documentos que têm referências a outros que foram redirecionados;
- Documentos que não podem ser acessados (ligações quebradas);

- Documentos com modificações recentes em seu conteúdo;
- Documentos com data de expiração próxima da data corrente.

O MOMspider acata as restrições definidas pelo responsável pelo servidor quanto a execução de robôs (arquivo robots.txt).O padrão para definição destas restrições foi apresentado por Martijn Koster em seu artigo Padrão para Exclusão de Robôs [KOS 94].

O acesso aos documentos dá-se através da requisição HEAD que transfere apenas as meta-informações existentes nos cabeçalhos, gerando com isso uma considerável redução de tráfego na rede e também não prejudicando a utilização normal do servidor.

O programa MOMspider pode ser obtido, na Internet, via anonymous ftp, no endereço:

ftp://liege.ics.uci.edu/pub/arcadia/MOMspider

3.3.3.2 URL-minder

O URL-minder é um serviço que percorre o WWW esporadicamente verificando se houve alguma mudança desde a última vez que este percorreu um determinado documento ou conjunto de documentos.

O serviço URL-minder é baseado em um formulário de cadastramento, onde o usuário informa quais páginas/documentos devem ser monitoradas pelo serviço. São verificados objetos acessíveis pelos protocolos HTTP (páginas WWW), FTP ou GOPHER.

3.3.3.3 Html_analyser

O objetivo do html_analyser é prover assistência a manutenção de bancos de estruturas WWW. As tarefas desempenhadas pelo html_analyser são as seguintes [PIT 95]:

- Extração de todas as âncoras, de todos os documentos HTML, da hierarquia do diretório. Os valores são extraídos tanto estando entre aspas (HREF=“ “) ou não (HREF=telnet).
- Criação de versões não-HTML dos documentos visando examinar as relações entre as ligações e o conteúdo das ligações.
- Validação da disponibilidade dos documentos destino das âncoras. (validate).
- Verificação do conteúdo das ligações existentes no banco de dados próprio com as existentes nos documentos (completeness).
- Verificação das relações um-para-um entre as ligações e o conteúdo das ligações (consistency).

O programa html_analyser pode ser obtido, na Internet, via anonymous ftp, no endereço:

ftp://ftp.cc.gatech.edu/pub/gvu/www/pitkow/html_analyser.

3.3.3.4 CHECKER

CHECKER é utilizado para verificar ligações em documentos HTML a procura de ligações inválidas.

O programa CHECKER está disponível para as seguintes plataformas:

- UNIX (HPUX, IRIX, SunOS, Linux, Solaris 2.4, FreeBSD, SCO)
- VMS
- Windows NT
- NeXT

A utilização do programa CHECKER dá-se a partir da linha de comando, fornecendo uma lista de opções e documentos HTML. Para especificar os documentos podem ser usadas expressões regulares, por exemplo: checker *.html.

O programa CHECKER pode ser obtido a partir do endereço:

http://www.ugrad.cs.ubc.ca/spider/q7f192/branch/checker.html.

3.3.3.5 Link Verifier

A motivação para o desenvolvimento do Link Verifier foi auxiliar aos responsáveis por servidores WWW na manutenção das ligações destes. O Link Verifier inicia analisando uma URL e, a partir desta, navega pelas ligações presentes segundo uma configuração de busca especificada previamente. Os resultados obtidos são utilizados para a geração de um relatório acerca do estado das ligações verificadas.

A ferramenta tem como desvantagem o fato de, por suas características de implementação, não verificar ligações com protocolos não-HTTP, isto é, ligações com GOPHER, FTP, TELNET.

São características do Link Verifier:

- Verificação configurável: O Link Verifier pode ser configurado pelo usuário ao especificar a URL inicial, o raio da verificação e restringir a verificação aos documentos locais, reduzindo, com isso, o “overhead” da verificação remota. A saída da verificação pode ser completa, onde todos as ligações são citadas, ou restrita a pontos com problema.
- Verificação paralela: Para a verificação de servidores remotos através de conexões de rede de baixa velocidade é implementada a verificação paralela. Esta consiste na execução de até 10 cópias locais do programa em servidores remotos.
- Verificação em “background”. Neste caso as saídas do programa serão enviadas ao responsável através de correio eletrônico.
- Verificação de formulários, através de ações “POST”.
- Uso consciente de banda de comunicação, implementado, entre outros, pela utilização de requisições HEAD (transferência somente do cabeçalho) ao invés de transferências completas de documentos, através do método GET.

O programa Link Verifier pode ser obtido a partir do endereço:

http://wsk.eit.com/wsk/dist/doc/admin/webtest/verify_links.html

3.3.3.6 Katipo

O programa Katipo procura por quaisquer documentos que tenha mudado desde a última vez que o usuário o acessou. Cria um relatório, o qual o usuário lê através de seu cliente WWW (Web Browser), listando estes documentos em um formato que facilita sua visita.

Katipo, a exemplo da maioria dos programa analisados, também implementa a característica de não transferir a totalidade dos documentos a serem analisados, transferindo apenas o cabeçalho.

A operação do Katipo é baseada na execução automática, diariamente, especialmente à noite, sendo que as mudanças são cadastradas e apresentadas no dia seguinte. A ferramenta Katipo somente verifica documentos indicados pelo protocolo HTTP, não tratando GOPHER, FTP ou TELNET.

Sendo um programa de domínio público, distribuído gratuitamente pela Internet, o Katipo está disponível para a plataforma Macintosh e, outras versões (Windows, UNIX, OS/2) não estão disponíveis nem planejadas.

O programa Katipo pode ser obtido a partir do endereço:

http://www.wuw.ac.nz/~newbery/Katipo.html.

3.3.3.7 ChURL

O programa ChURL, um script PERL versão 4, foi definido para verificar a validade das ligações em documentos HTML. São disponibilizadas várias opções, incluindo a habilidade da verificação recursiva de documentos por servidores WWW.

ChURL têm duas utilizações significativas:

- Individual: Tipicamente utilizada por usuário para a verificação de “home-pages” pessoais.
- Corporativa: Utilizada por administradores de servidores WWW, realiza uma verificação completa da hierarquia do servidor.

A saída do programa é exibida na console do computador e é constituída basicamente de códigos de erro do HTTP, além de algumas mensagens de erro próprias.

O programa ChURL pode ser obtido através do endereço:

http://www-personal.engin.umich.edu/~yunke/scripts/churl/.

3.3.3.8 Checkbot

Checkbot é uma ferramenta que analisa um conjunto de páginas existentes em uma hiperbase e verifica todas suas ligações.

O programa é inicializado com dois argumentos principais: URL de início, onde a verificação iniciará e "Site String", que identifica univocamente um servidor. Caso o endereço de alguma das páginas analisadas não coincidir com a "Site String", a princípio, a página não será analisada.

A operação do Checkbot é baseada na coleta de todas as ligações do servidor que coincidem com a "Site String", e que são acessáveis via URL de início. As ligações são divididas em duas categorias: Internas e Externas. O processo é iniciado pela verificação das ligações internas, sendo tratadas após este, as ligações externas.

O programa Checkbot pode ser obtido através do endereço:

http://dutifp.twi.tudelft.nl:8000/checkbot-info.html.

3.3.3.9 Astra Site Manager

A Astra SiteManager é uma ferramenta visual de gerenciamento de hiperbases, desenvolvida para ambientes Windows 95 e NT, projetada para servidores que apresentem rápida velocidade de crescimento e grandes alterações de conteúdo.

São características do Astra SiteManager:

- Criar representações gráficas de toda a hiperbase;
- Verificar e corrigir inconsistências de ligações usando o módulo "Link Doctor";
- Acompanhar mudanças na hiperbase através do módulo "Change Viewer"
- Construir rotinas específicas através da "Astra's Application Program Interface (API)"

Astra SiteManager é composto dos seguintes módulos:

- Análise de Mapas - Através do módulo "Fast Scan Web Site Mapping". Este módulo apresenta uma grande capacidade de varredura paralela e geração gráfica de mapas. O processo rapidamente percorre toda a hiperbase e monta uma representação gráfica, codificada através de cores específicas. Durante o processo de mapeamento a ferramenta detecta dinamicamente mudanças na estrutura da hiperbase. Pode-se afirmar que o Astra SiteManager é a única ferramenta capaz de gerenciar e mapear tanto ligações internas quanto externas, bem como documentos estáticos e dinâmicos.
- Análise de Visões Estruturais - Através do módulo "Visual Web DisplayTM". Este apresenta a capacidade de construir representações hierárquicas, em formato de estrutura de árvores, diminuindo com isso a complexidade da análise de toda uma hiperbase. Além disso, a ferramenta pode ser configurada para análises específicas, como por exemplo: Programas CGI ou somente ligações internas à hiperbase. São disponibilizados recursos de "ZOOM-IN" e "ZOOM-OUT", foco instantâneo,

manipulação de janelas e impressão para análise “off-line”. Na figura 3.5 é apresentada a visão estrutural de uma hiperbase gerada por este módulo.

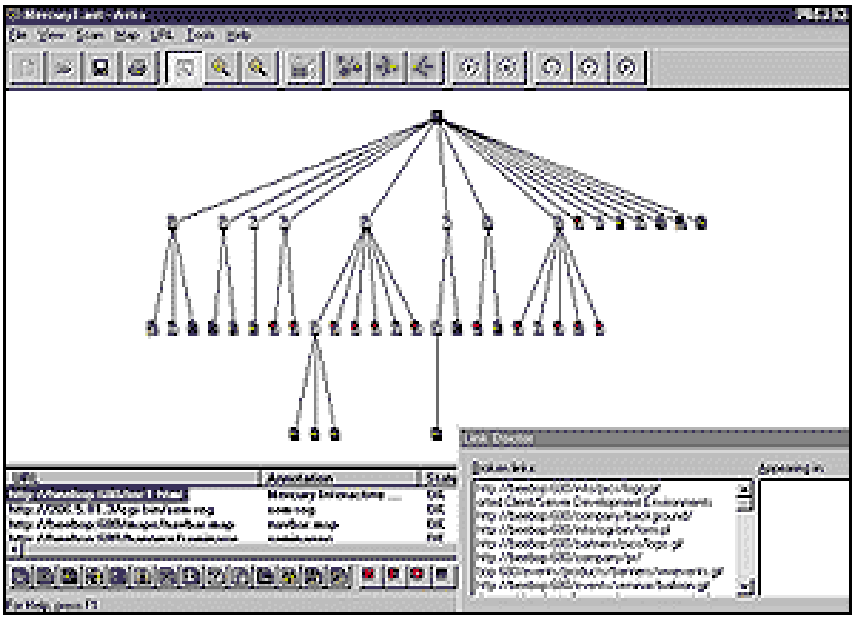

Fig. 3.5. Visão estrutural de uma hiperbase gerada pelo Visual Web DisplayTM

- Análise de Padrões - Através do módulo “Action TrackerTM”, a ferramenta é capaz de, graficamente, apresentar padrões de utilização para análise e otimização da hiperbase. O Astra SiteManager transforma os arquivos de “log“ gerados pelos servidores HTTP em mapas com formato de hiperdocumento, provendo uma visão gráfica e colorida da movimentação dos usuários pela hiperbase. Utilizando consultas, os responsáveis pela manutenção da hiperbase podem analisar como os usuários navegam pelos documentos, identificando os caminhos mais utilizados e aqueles que não tem apresentado interesse. Nas figuras 3.6 e 3.7 são apresentados, respectivamente, a legenda do módulo “Action Tracker” e um exemplo de análise de movimentação gerado.

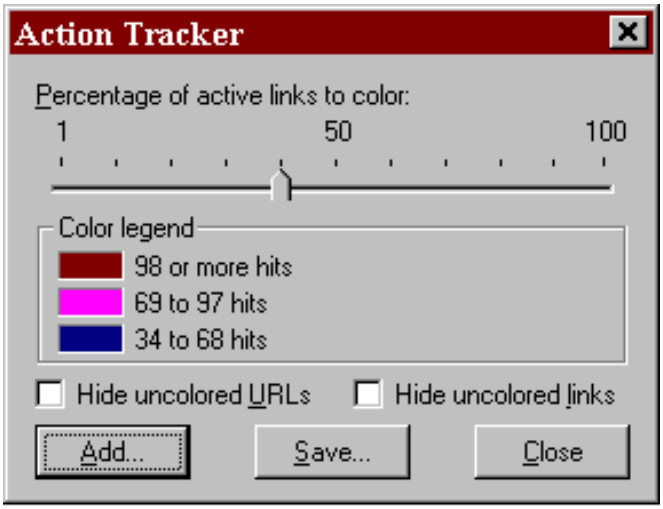

Fig. 3.6 Módulo “Action Tracker”

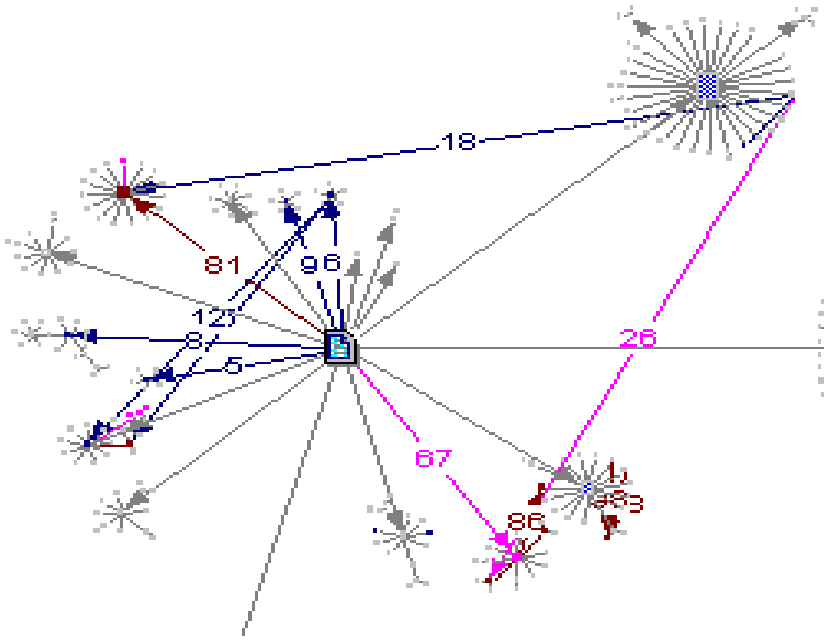

Fig 3.7 Exemplo de análise de movimentação, gerado pelo módulo "Action Tracker"

- Análise de estrutura - Através do módulo "Change ViewerTM". Este compara dois diferentes momentos da arquitetura da hiperbase , monitorando todas as alterações efetuadas no período. As comparações podem incluir documentos e ligações adicionados, alterados e mesmo excluídos.

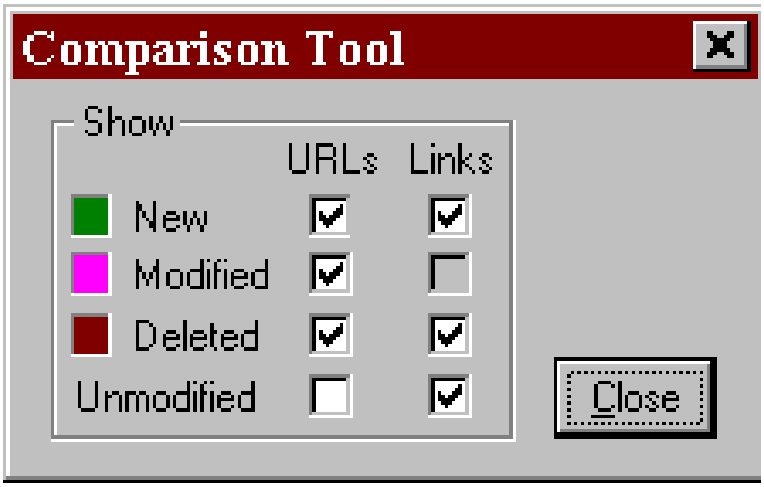

Fig 3.8. Módulo Change ViewerTm

- Análise Avançada de Ligações - Através do módulo "Link DoctorTM". A ferramenta pode identificar e exibir todos os problemas relacionados com as ligações. Por exemplo: Todos as ligações que apresentam inconsistência são exibidos em cor vermelha para uma fácil detecção. Também é possível uma análise individualizada por determinadas classe de erros (erro 404, erro 403).

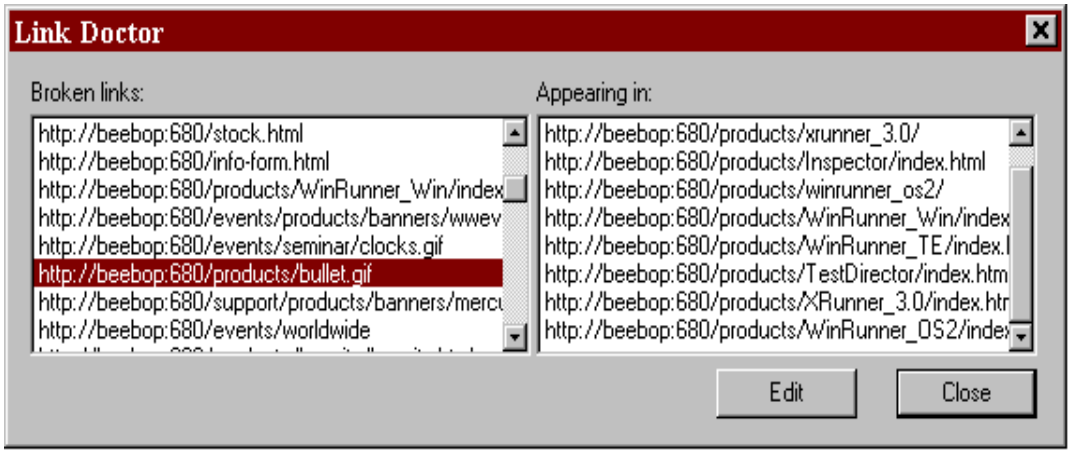

Fig. 3.9. Ligações inconsistentes apresentadas pelo módulo " Link Doctor " .

- Análise de documentos e ligações dinâmicos - Através do módulo "DynamicScanTM" é possível analisar não somente ligações e documentos estáticos mas também ligações e documentos gerados dinamicamente, que representam informações contidas em banco de dados ou geradas em tempo real. Uma aplicação importante dessa característica é a possibilidade, pelos responsáveis pela hiperbase, de analisar documentos gerados por sistemas de "home banking", certificando-se com isso que os correntistas estão realmente recebendo as informações solicitadas.
- Capacidade de Expansibilidade - Através da utilização da "Astra's Application Program Interface (API)", responsáveis pela hiperbase podem facilmente construir "plug-ins", como analisadores de 'log" e mecanismos de busca, visando estender a capacidade da ferramenta. São disponibilizadas rotinas em linguagem Java, C++ e Visual Basic.

A ferramenta pode ser obtida em http:// www.merc-int.com/products/astraguide.html

3.3.3.10 InfoLink Link Checker

A ferramenta visual InfoLink Link Checker, desenvolvida pela empresa BiggByte Software's, disponível na plataforma Windows 95 e NT, fornecida em versões "freeware" e comercial. Tem por objetivo ser uma ferramenta de pesquisa pequena, de fácil e rápida utilização.

Seus objetivos restringem-se a servidores pequenos, com baixa taxa de crescimento e pouca complexidade. Pela sua facilidade de uso e características simples, sua utilização é indicada para a manutenção de integridade de hiperbases pessoais.

São características da ferramenta:

- Verificação de todos os tipos de ligação HTML;
- Suporte a janelas visuais;
- Programa de navegação incorporado;
- Programa de edição de documentos HTML incorporado;

- Análise de ligações inconsistentes;
- Visualização em estrutura de árvores.
- Verificação de páginas internas e externas;
- Suporte a servidores “proxy”
- Verificação em servidores FTP;
- Verificação de “DNS lookup”;
- Procura por palavra-chave;

A ferramenta pode ser obtida em http:// www.bigbyte.com.br

3.3.3.11 Linkbot

O LinkBot é um conjunto completo de ferramentas de gerenciamento de hiperbases. Este conjunto abrange todos os módulos necessários à automatização da manutenção de hiperbases, mantendo-a livre de erros.

Elaborada para o ambiente Microsoft Windows 95, possibilita ao responsável o acompanhamento e a reparação dos problemas que possam ocorrer na hiperbase.

A figura 3.10 apresenta um exemplo de visualização da estrutura e organização de uma hiperbase utilizando a ferramenta LinkBot.

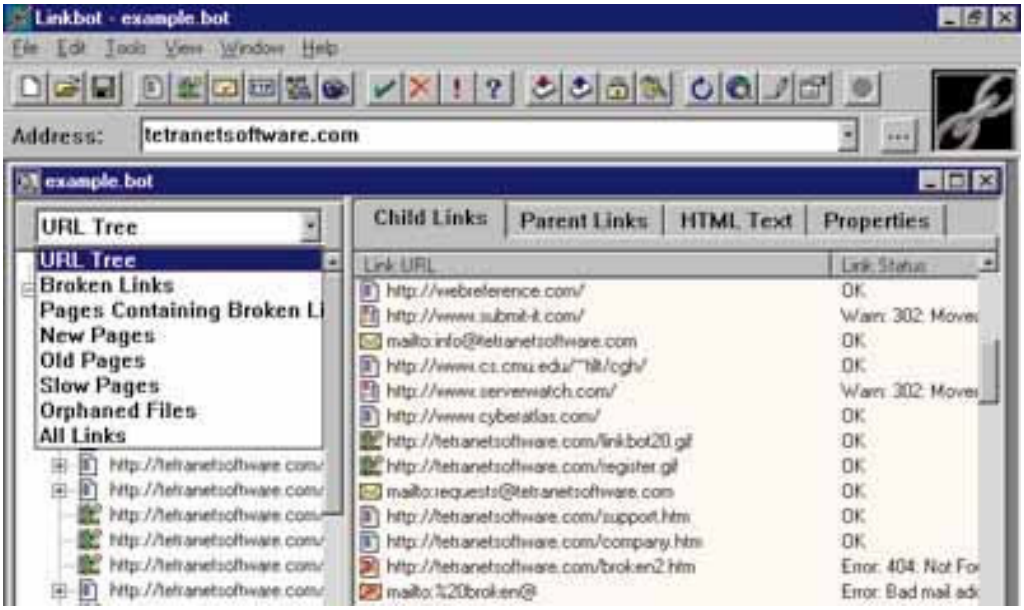

Fig. 3.10 Exemplo de visualização da estrutura e organização de uma hiperbase utilizando a ferramenta LinkBot.

São características da ferramenta LinkBot:

- Gerenciamento de inconsistências de ligações, imagens e conteúdos - Partindo das afirmações que a qualidade de uma hiperbase é uma direta reflexão da empresa que o produziu, que erros como inconsistências de ligações ou conteúdo desatualizado geram uma impressão negativa sobre a empresa e seus produtos, a ferramenta auxilia a manter uma hiperbase livre de erros através de uma rápida detecção e ajuda na correção destes.
- Identificação de arquivos não utilizados (órfãos) - Procura auxiliar na decisão sobre remover ou não ou objeto da hiperbase. O programa perfaz uma busca em toda a hiperbase por documentos que não são mais utilizados, reportando-os ao gerente desta.
- Procura por documentos com grande tempo de "download": Sabe-se que atualmente 70% dos usuários Internet utilizam conexões a 28.8 kbs ou menos. Baseado nisso, pode-se afirmar que a hiperbase certamente estará perdendo visitantes, devido a altos tempos de recuperação de documentos.
- Visualização da estrutura e organização da hiperbase: Permite criar um mapa da hiperbase através de uma interface intuitiva.
- Organização da manutenção de grupos de trabalho: Caso se tenha mais de uma pessoa desenvolvendo o conteúdo da hiperbase, a ferramenta pode ser customizada para emitir relatórios que separam os problemas por autor de cada página.
- Escalonação de análises e rotinas de manutenção: Permite incrementar a consistência de uma hiperbase através de análises e manutenções automáticas periódicas.

A tabela 3.1 apresenta uma relação de funcionalidades deste conjunto de ferramentas e dos benefícios que, se espera, sejam decorrentes destes.

TABELA 3.1 Relação das Funcionalidades da Ferramenta LinkBot

Funcionalidade	**Benefício**
Núcleo multi-tarefa de alta performance	Permite analisar hiperbases complexas de maneira extremamente rápida
Verificação de ligações internas e externas dos tipos HTTP, HTTPs e FTP	Permite isolar todos as ligações inconsistentes em uma hiperbase
Verificação de documentos não utilizados (órfãos)	Permite eliminar seguramente todos os documentos dispensáveis.
Verificação de documentos com conteúdos obsoletos	Permite identificar documentos que podem estar obsoletos.
Verificação de documentos com altos tempos de recuperação	Permite identificar e modificar documentos que possam estar prejudicando a recuperação da hiperbase.
Verificação de documentos quanto a sintaxe das ligações tipo "mailto"	Permite identificar e corrigir documentos que apresentem ligações do tipo "mailto" inválidas

Funcionalidade	Benefício
Verificação de documentos com falta de título	Permite identificar e reparar documentos que não apresentam título (marca <TITLE>.
Utilização de filtros para especificação de documentos	Permite especificar documentos ou áreas da hiperbase a serem analisadas
Utilização de interface familiar e intuitiva	Permite ao responsável pelo gerenciamento de hiperbase explorar esta através de uma interface familiar.
Exibição de ligações que entram e saem de uma determinada URL	Torna a correção de ligações inconsistentes facilitada pela exibição das ligações referentes
Classificação de documentos por características especiais	Permite isolar documentos com problemas específicos
Disponibilização de filtros avançados	Permite exibir de pequenos subconjuntos de uma hiperbase
Editor HTML incorporado	Possibilita editar e corrigir internamente a interface do programa
Pré-checagem de conjuntos de URLS	Permite agir sobre conjuntos especificados de URLs, por exemplo todas as ligações inválidas a hiperbases externas
Exportação dos resultados de análises	Permite armazenar os resultados em arquivos delimitados ou em banco de dados
Criação de nove tipos diferentes de relatórios	Apresenta um sumário dos problemas da hiperbases em relatórios detalhados de ligações inválidas, documentos inválidos, avisos de "warning", documentos órfãos, novos, velhos e grandes.
Criação de relatórios personalizados por autor	Facilita o trabalho em grupo
Criação manutenções programadas	Permite incrementar a consistência de uma hiperbase através de análises e manutenções automáticas regulares
Suporte ao padrão de exclusão de robôs (robots.txt)	Permite que o programa não analise determinadas áreas de uma hiperbase
Suporte ao padrão de "cookies"	Permite que o programa simule visualizadores que aceitam "cookies"
Análise hiperbases locais ou remotas	Permite analisar hiperbases hospedadas em servidores não-windows.
Suporte a servidores "proxies" e requisições autenticadas	Permite analisar ligações externas à "firewalls" e páginas que requerem identificação de usuário

A ferramenta pode ser obtida em http:// www.tetranetsoftware.com .

3.3.3.12 LinkScan

A ferramenta LinkScan, desenvolvida pela Electronic Software Publishing Corporation (Elsop), para ambientes MS-Windows e UNIX, tem a capacidade de detectar automaticamente inconsistências de ligação causadas por documentos desaparecidos ou URL não disponíveis.

Estudos efetuados pela empresa mostram que a maioria das hiperbases apresentam algum tipo de problema. Estes incluem até 30% de ligações externas com inconsistência, documentos desatualizados ou órfãos. A solução apresentada baseia-se em um poderoso e automático validador de ligações capaz de testar cada ligação na hiperbase.

São características da ferramenta LinkScan:

- Procura por documentos HTML, imagens e outros arquivos;
- Validação de todas as ligações internas;
- Verificação de todos os nomes de referências e marcas;
- Criação de dois tipos de mapas das hiperbases;
- Verificação de documentos órfão;
- Validação de ligações externas;
- Verificação de problemas de DNS e documentos não-autorizados;
- Verificação de documentos expirados;
- Performance de teste de até 40.000 ligações/hora;
- Capacidade de validação em hiperbases com mais de 50.000 ligações externas;
- Capacidade de validação em hiperbases com mais de 80.000 ligações internas;
- Processamento simultâneo permite checagem de 60 ligações concorrentemente;
- Suporte a “aliases” de servidores e redirecionamentos;
- Suporte a servidores “proxy”;
- Suporte a documentos dinâmicos, criados por “CGIs”, “ASPs”, e “RDBMSs”;
- Criação de arquivo-histórico de problemas com ligações externas para acelerar processamento;
- Controle avançado para evitar a checagem de ligações duplicadas;
- Controle de reinicialização para problemas ocorridos durante os testes;
- Controle sobre periodicidade de manutenção de ligações externas;
- Possibilidade de escalonamento de processos de manutenção;
- Possibilidade de manutenção em documentos protegidos por senhas;
- Características de gerenciamento multi-servidor;
- Manutenção preventiva, tornando possível o redirecionamento de ligações;

- Geração de relatórios em formato HTML, com ligações incluídas;
- Operação em todos as variantes de UNIX;
- Operação em Microsoft WindowsNT 4.0;
- Geração de mapas de hiperbases (até 400 documentos);
- Facilidade de instalação e operação;
- Documentação em formato HTML.

A ferramenta pode ser obtida em http:// www.elsop.com.

3.3.3.13 Netscape LiveWire

Netscape LiveWire e LiveWire Pro são conjuntos de ferramentas visuais, desenvolvidos pela empresa Netscape Corporation, disponíveis paras as plataformas Windows95 e Windows NT, que atuam no gerenciamento de hiperbases e geração de conteúdo dinâmico.

São compostos por:

- Netscape Navigator Gold;
- LiveWire Site Manager;
- LiveWire JavaScript Compiler;
- LiveWire Database Connectivity Library.

Destas, o módulo LiveWire Site Manager foi desenvolvido com o objetivo de ser uma ferramenta visual para gerenciamento de hiperbases com facilidades de técnicas de "drag-and-drop".

As características do LiveWire Site Manager são:

- Gerenciamento visual, que provê uma visão gráfica da organização da hiperbase;
- Tecnologia de reestruturação, através de reogranização automática de mudanças em todas as referências a um documentos, ligação ou arquivo, desde que estes objetos tenham sido construídos com ferramentas do Netscape LiveWire;
- Validação de ligações externas;
- Disponibilização de processos assistentes, que conduzem usuários novatos por um processo bem organizado de autoria;
- Capacidade de importação de hiperbases completas para análise, edição e publicação local;
- Processo de publicação de hiperbases simplificada (tipo um-toque);
- Conversores de arquivos gráficos, incluindo BMP, WMF, PCX e outros;
- Compilador JavaScript incorporado.

A ferramenta pode ser obtida em http://www.Netscape.com/comprod/server_central/product/livewire/livewer_datashet.html.

3.3.3.14 SiteSweeper

Devido ao crescimento acelerado da Internet e das Intranet corporativas tornou-se função dos profissionais responsáveis pelo gerenciamento de hiperbases a manutenção da integridade de documentos e aplicativos de missão-crítica baseados em web. O SiteSweeper, desenvolvida pela empresa Site technologies inc., busca manter um controle de qualidade em hiperbases profissionais, provê meios para que esta tarefa seja realizada de maneira mais eficaz através do gerenciamento das informações necessárias.

Sabe-se que documentos com grandes imagens são a maior fonte de frustação por parte dos usuários. Baseado neste fato, o SiteSweeper automatiza o processo de determinação do tamanho total de todos os documentos da hiperbases e apresenta análises baseadas em diversos tipos de conexão possíveis.

A verificação de inconsistência de ligações também é automatizada pelo SiteSweeper, ajudando o responsável a imediatamente identificar e resolver estes problemas antes que eles se alastrem por toda hiperbase. Os tipos de inconsistências identificadas como “404 - Arquivo não encontrado”, “401 - Não autorizado“ ou “301 - movido permanentemente” são identificadas e descritas nos relatórios gerados.

As informações providas pelo SiteSweeper são:

- Propriedades de cada documento;
- Estatísticas de servidores agregados;
- Ligações “de” e “para” cada documento;
- Ligações para documentos externos;
- Catálogos/Imagens reduzidas de todas as imagens da hiperbase.

A utilização da ferramenta SiteSweeper é bastante simples e intuitiva. Normalmente dá-se através do preenchimento do endereço da hiperbase a ser analisada. A partir disto, o processo percorre todos os documentos e ligações, verificando a integridade interna e externa. Após o término do processo são gerados uma série de relatórios em formato HTML.

A lista de relatórios disponíveis inclui:

- Tamanho total de cada documento;
- Documentos com inconsistências de ligações;
- Catálogo de imagens;
- Comentários sobre cada documento.

As características da ferramenta SiteSweeper são:

- Relatórios detalhados por demanda ou escalonamento;
- Suporte a múltiplos servidores;

- Suporte a autenticação em documentos protegidos por senhas;
- Processamento paralelo para incremento de performance;
- Suporte a servidores “Proxy”.

Os requisitos de hardware e software para o uso do SiteSweeper (programa analisador) são:

Windows 95 ou Windows NT;

Processador Intel i486 ou superior;

Memória RAM de 8MB (para Windows95) e 16Mb (para Windows NT).

A ferramenta pode ser obtida em: http://www.sitetech.com/sitesweeper.

3.3.3.15 WebMapper

A ferramenta WebMapper 2.0, desenvolvida originalmente pela empresa NetCarta e posteriormente incorporada aos servidores da Microsoft, visa auxiliar a verificação de hiperbases que apresentam crescimento acelerado, dificultando a manutenção e propiciando a incidência de erros 404.

A ferramenta é baseada na tecnologia de “spiders” - é preciso somente informar o endereço do documento principal da hiperbase para que se comece a validar toda esta. O esquema de funcionamento é equivalente ao utilizado por ferramentas de busca e diretórios web, como o “Lycos” e “Infoseek”, mas ao invés de varrer toda a Internet, restringe-se à apenas um pequeno subconjunto.

O resultado da validação é exibido por uma mapa da hiperbase acompanhado por relatórios detalhados que descrevem claramente os problemas encontrados, como documentos órfão, ligações inconsistentes e imagens não encontradas.

Os mapa são gerados a partir de uma abordagem inovadora. Além dos mapas usuais, hierárquicos, é apresentado um mapa “Cyberbolic”, que pode ser definido essencialmente como uma visão olho-de-peixe. Um exemplo de mapa “Cyberbolic” pode ser vista na figura 3.11.

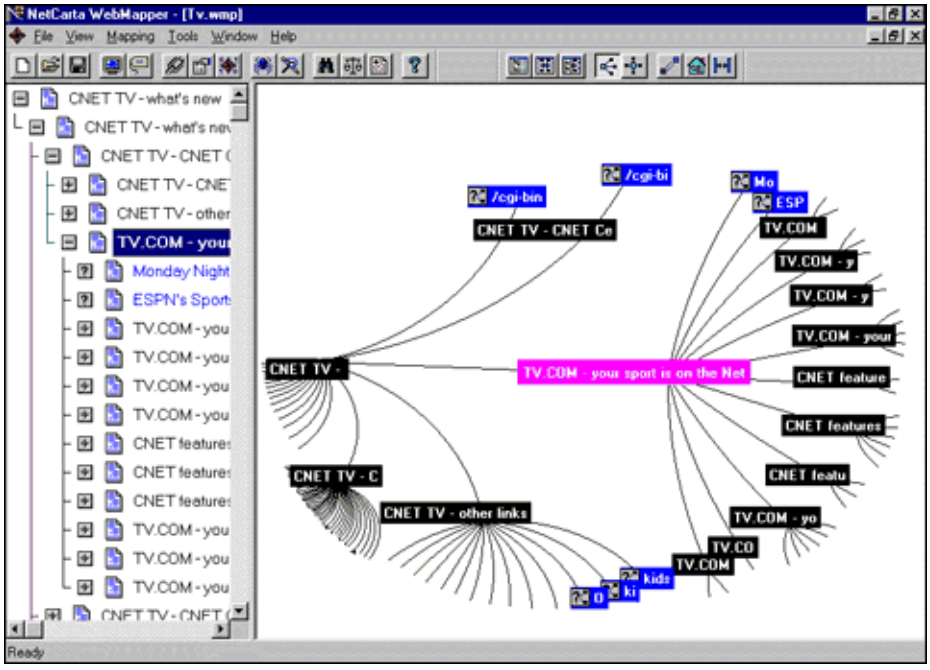

Fig. 3.11 - exemplo de mapa "Cyberbolic"

Os requisitos de hardware e software para a ferramenta WebMapper são:

- Processandor Intel i486 ou superior;
- Memória de 8Mb RAM;
- Espaço de ao menos 7 Mb no disco rígido;
- Windows 95 ou Windows NT 3.51.
- Visualizador WWW.

A ferramenta pode ser obtida em http:// www.netcarta.com.

4 Solução Proposta e Protótipo Implementado

Visando apresentar plenamente a solução proposta e protótipo implementado, serão abordados sua definição, objetivos, classificação, estrutura, interface e características de implementação.

4.1 Definição e Funcionamento

A solução proposta tem por objetivo minimizar o problema da inconsistência de ligações no modelo WWW junto ao servidor do Instituto de Informática da UFRGS.

Atuando de forma preventiva semi-automática, a ferramenta desenvolvida apresenta um novo enfoque de tratamento ao problema da inconsistência de ligações, baseando-se em formulários HTML e banco de dados de ligações. A solução é implementada em nível de hiperbase local, sendo que a distribuição dar-se-á através da replicação da solução em diferentes hiperbases.

O enfoque preventivo ocorre quando procura evitar a inconsistência, através do cadastramento de novas localizações válidas. O cadastramento de ligações é realizado através do preenchimento de um formulário específico pelo proprietário - ou terceiro autorizado - dos documentos em questão.

O enfoque semi-automático é contemplado quando da ocorrência do tratamento de um erro, devido ao acesso a um documento com localização inválida (Erro 404 - Arquivo Não Encontrado). A partir dos dados previamente cadastrados, a ferramenta é capaz de redirecionar a requisição de forma automática e transparente ao usuário.

Pode-se destacar como vantagens deste tipo de solução ou abordagem:

- Integridade: Mantém a integridade do servidor em nível de acessos externos;
- Descentralização: Não necessidade de intervenção direta do administrador do servidor para cada ocorrência de atualização;
- Praticidade: Possibilidade de múltiplos redirecionamentos;
- Transparência: Processo de redirecionamento automático;
- Extensibilidade: Integração com ferramentas já implementadas no servidor;
- Segurança: Não possibilidade de redirecionamento de documentos válidos, além da utilização de senhas para documentos particulares;
- Desempenho: Baixo “overhead” no processamento das requisições pelo servidor.

Podem ser consideradas desvantagens desta abordagem:

- Necessidade de cadastramento manual de alterações;
- Necessidade de cadastro individual para cada documento.

O funcionamento da ferramenta baseia-se nas seguintes etapas:

- Cadastramento prévio, pelos usuários, de alterações de localização de documentos, através de um formulário disponibilizado através da página principal;

- Consulta a este cadastro quando da ocorrência de erros de ligação (erro 404) e posterior redirecionamento da requisição para a nova localização.

A estrutura da ferramenta pode ser vista na figura 4.1:

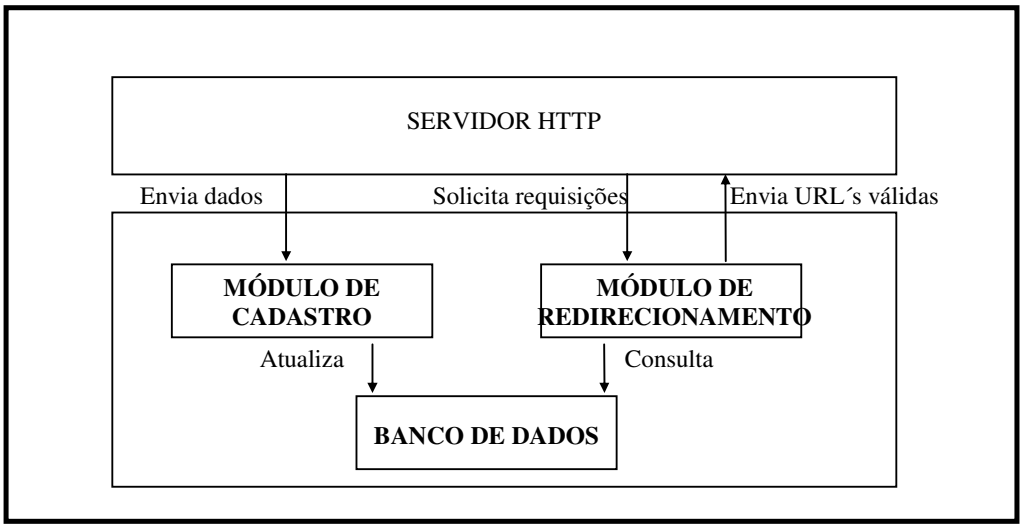

FIGURA 4.1 - Estrutura da ferramenta

A ferramenta é composta de dois módulos:

- Módulo de Redirecionamento;
- Módulo de Cadastro.

O Módulo de Cadastro é responsável pelo armazenamento em um banco de dados específico, das informações fornecidas através do preenchimento do formulário de cadastro.

O formulário disponibilizado aos usuários pode ser visto na figura 4.2:

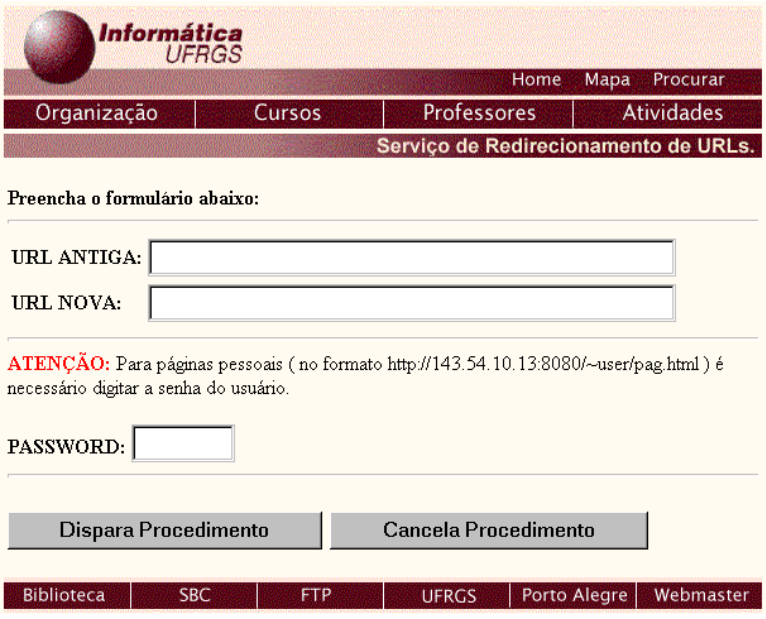

FIGURA 4.2 - Formulário de cadastro de URLs

O funcionamento do Módulo de Cadastro dá-se através das seguintes etapas:

1. O usuário preenche o formulário HTML com os dados do redirecionamento;
2. O Módulo de Cadastro recebe os dados do formulário;
3. Caso a localização anterior de um documento seja em um diretório particular de usuário o programa confere a senha digitada.
4. Caso a senha digitada não confira com a armazenada no arquivo de senhas do sistema operacional, o programa monta uma página de resposta informando ao usuário que este não possui direito de alterar páginas daquele usuário. A página de resposta gerada pode ser vista na figura 4.3:

FIGURA 4.3 - Página reportando senha inválida

5. Caso a senha digitada seja válida, o programa consulta o banco de dados a fim de verificar se a localização anterior já está cadastrada. Se existe cadastro, é efetuada uma alteração no registro correspondente, senão é criado um novo registro.
6. Após a inclusão do novo registro no banco de dados é exibida uma página em formato HTML que informa ao usuário que o procedimento foi completado com sucesso. Um exemplo desta página pode ser vista 4.4:

FIGURA 4.4 - Página reportando sucesso no procedimento

O Módulo de Redirecionamento é responsável por tratar as ocorrências do erro 404 geradas pelo servidor. Para tanto, este recebe o indicativo de erro gerado pelo servidor e realiza uma consulta ao banco de dados de localizações.

O funcionamento do Módulo de Redirecionamento dá-se através das seguintes etapas:

1. O visualizador envia ao servidor uma requisição de documento;
2. O servidor recebe a requisição e executa uma busca na árvore de diretórios pelo arquivo solicitado;
3. Caso o servidor encontre o arquivo solicitado ele o retorna normalmente ao visualizador;
4. Caso o arquivo não seja encontrado o servidor gera uma mensagem de erro HTTP 404 - file not found, que aciona o módulo de redirecionamento (a mensagem *não* é repassada ao visualizador/cliente);
5. O módulo de redirecionamento efetua uma consulta ao banco de dados de ligações buscando a nova localização do documento requerido.
6. Caso a nova localização esteja cadastrada no banco de dados, o programa envia uma mensagem especial com a nova localização para o servidor que, então, devolve o documento para o visualizador.
7. Caso a nova localização não esteja cadastrada, o módulo de redirecionamento reporta o fato ao usuário e indica possibilidades de localização através da ferramenta de busca disponibilizada no servidor. A página da ferramenta de busca pode ser vista nas figuras 4.5 e 4.6:

FIGURA 4.5 - Página indicando documento não localizado

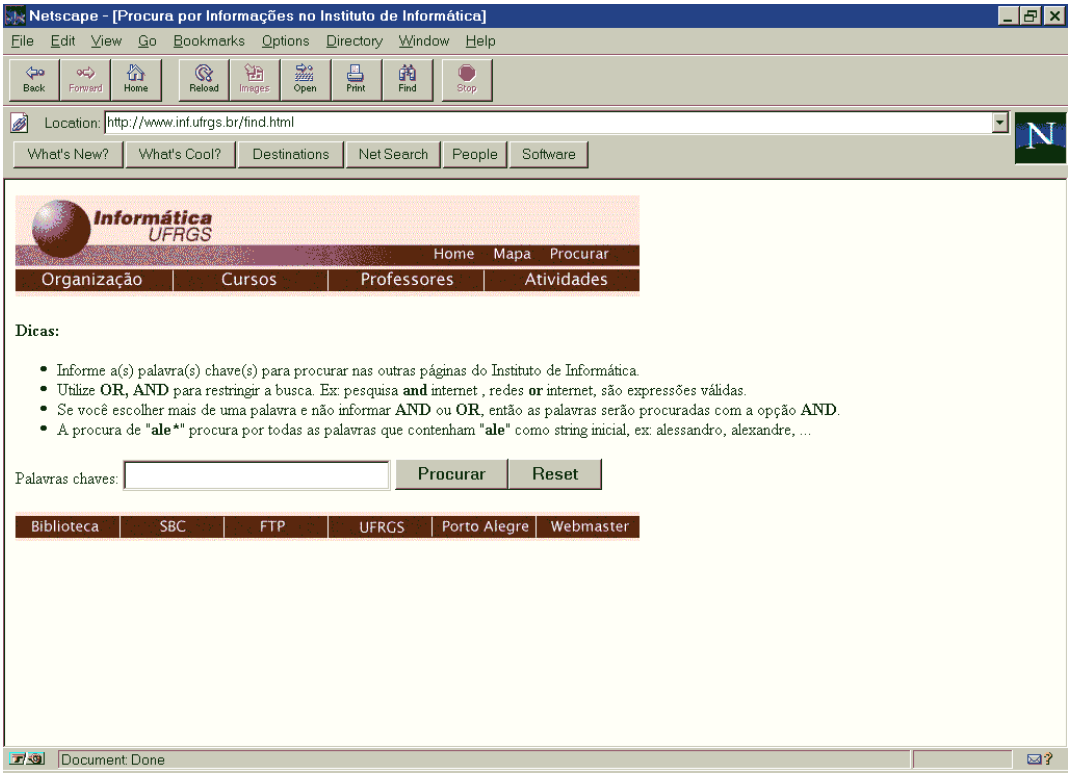

FIGURA 4.6 - Página exibindo ferramenta de busca

4.2 Características de Implementação

O protótipo foi implementado através de dois programas CGI (Common Gateway Interface), codificados em linguagem C e localizados no diretório CGI-BIN do servidor HTTP.

O armazenamento das localizações (URLs) é efetuado em um banco de dados, implementado através de um arquivo-texto, denominado cadastro.dat, também localizado no diretório CGI-BIN.

A estrutura do arquivo cadastro.dat pode ser vista na figura abaixo:

CADASTRO.DAT		
URL Antiga	separador (TAB)	URL Nova

FIGURA 4.7 - Estrutura do arquivo de dados

4.2.1 Módulo de Cadastro

O modelo de cadastro, implementado através do programa CADASTRO.C é executado a partir de um formulário HTML de cadastro (cadastro.html) através do comando:

```
<form method=POST action="/cgi-bin/cadastro.cgi">
```

Os dados recebidos são enviados pelo formulário através do método POST.

A senha fornecida é conferida com o arquivo *.passwd* do UNIX através da função confere_senha, que recebe como parâmetros o nome e a senha do usuário. O nome do usuário, que não é fornecido pelo formulário, é obtido através da string que contém a localização antiga, como na figura abaixo:

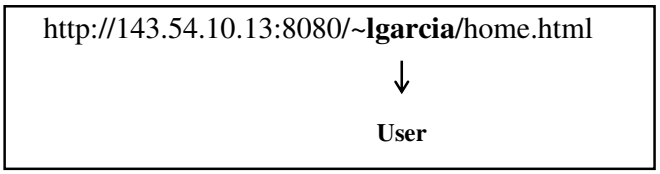

FIGURA 4.8 - Identificação do usuário

A senha é fornecida através de um campo do formulário. O conjunto nome-senha é comparado com o armazenado no arquivo através da função **crypt**.

O armazenamento dos dados é realizado pela função **grava**, que opera sobre dois arquivos texto:

- cadastro.dat: Arquivo que armazena os dados das localizações;

- temp.dat: Arquivo temporário.

Caso a abertura ou fechamento dos arquivos de dados apresente problemas são exibidas páginas HTML reportando o fato ao usuário que deve, então, avisar o administrador do sistema.

Na função **grava** é realizada a pesquisa para verificar se a localização já esta cadastrada e caso esteja a função apenas atualiza o registro. Caso não esteja cadastrada é criado um novo registro no final do arquivo. Os registros são armazenados através de linhas no arquivo texto utilizando como separador o caracter de TAB.

4.2.2 Módulo de Redirecionamento

O módulo de redirecionamento, implementado através do programa TRATA404.C é executado pelo servidor quando da ocorrência de um erro 404.

A diretiva que indica ao servidor que caso ocorra erro tipo 404 o processamento deva ser desviado para o programa trata404.cgi é armazenada no arquivo srm.conf:

```
ErrorDocument 404 /cgi-bin/trata404.cgi
```

O módulo de redirecionamento recebe, via estas variáveis de ambiente, informações sobre a máquina, porta e localização requisitada:

- SERVER_NAME;
- SERVER_PORT;
- REDIRECT_URL.

Estas variáveis são especiais e somente são disponibilizadas pelo servidor quando da ocorrência de erros da classe 400.

O redirecionamento é realizado através do cabeçalho **Location:**, como visto no trecho abaixo:

```
printf("Status: 302 Redirect\r\n");
printf("Location: %s\n\n",nova_localizacao);
```

Outra alternativa para o programa CGI seria através de comandos JavaScript, como visto no trecho abaixo:

```
printf("Content-type: text/html%c%c",10,10);
printf("<HTML>");
printf("<HEAD>");
printf("<SCRIPT Language=\"JavaScript\">");
printf("window.location = %s\n\n",nova_localizacao);
printf("</SCRIPT>");
printf("</HEAD>");
printf("<BODY>");
printf("</BODY>");
printf("</HTML>");
```

4.3 Ambiente de Prototipação

O servidor utilizado foi o NCSA versão 1.5.2a, rodando sobre sistema operacional SUN OS em uma estação de trabalho SUN compatível.

O servidor utilizado para a prototipação, desenvolvimento e validação foi instalado em uma conta particular de usuário e recebeu o endereço http://143.54.10.13:8080, onde 143.54.10.13 corresponde ao endereço IP da máquina (sagui.inf.ufrgs.br) e 8080 a porta utilizada.

Foi utilizada a porta 8080 para diferenciar da porta padrão para serviços HTTP, porta 80, evitando assim que, inadvertidamente, usuários externos pudessem utilizá-lo sem conhecimento de sua natureza de testes.

O servidor pode ser executado através da seguinte linha de comando: httpd -d /home/coruja/lgarcia/httpd, a partir da máquina sagui.inf.ufrgs.br.

5 Conclusões e Trabalhos Futuros

Com a popularização da Internet, dos Sistemas de Hiperdocumentos Distribuídos e principalmente da World-Wide Web tornou-se necessário que se considerasse seriamente o problema da inconsistência de ligações entre os recursos disponíveis.

Ao longo desta dissertação buscou-se caracterizar este problema nos diferentes Sistemas de Hiperdocumentos Distribuídos existentes na bibliografia, com enfoque especial ao ambiente WWW, que, pela popularidade e disponibilidade tornou-se objeto de pesquisa e desenvolvimento do protótipo da solução proposta.

Por ser o WWW um ambiente que não provê soluções preventivas para este problema, a solução encontrada foi desenvolver uma ferramenta que atuasse como Solução Preventiva que procura atenuar o problema.

Pode-se considerar como a principal contribuição apresentada pela ferramenta o tratamento da chamada Integridade Externa, acompanhada da prevenção, descentralização, praticidade, transparência, extensibilidade, segurança e alto desempenho.

Percebeu-se que, com o uso efetivo da ferramenta obteve-se uma melhora no aspecto da Consistência de Ligações em WWW. Esta, porém, ainda não provê uma solução totalmente eficaz no momento em que é classificada como semi-automática, requerendo, ainda que pouca, intervenção humana.

A evolução da ferramenta para totalmente preventiva automática, sendo que a cada movimento de arquivos no servidor estes sejam registrados automaticamente no banco de dados de ligações, sem depender da intervenção dos usuários, é um tema para estudo futuro que viria a efetivamente resolver o problema, mantendo o servidor ainda mais íntegro. A automatização do cadastramento poderia ser feita via monitoração do sistema de arquivos de todas as estações de trabalho da rede ou através de robôs/agentes que poderiam percorrer tanto a rede interna como servidores externos quando desejado (e permitido).

Pode-se considerar como fator limitante da ferramenta a questão do formulário de cadastro de localizações apresentar somente a possibilidade de redirecionar um documento por vez, o que pode ser considerado pouco produtivo. Permitir o redirecionamento de árvores de diretórios pode ser outro tópico a ser contemplado em uma implementação futura. A possibilidade de comunicação entre cópias replicadas da solução em diferentes hiperbases seria, também, uma funcionalidade a ser acrescentada em versões posteriores.

Considerou-se aqui como solução todo o processo de identificação, análise, comparação, pesquisa bibliográfica, prototipação, validação e descrição de uma técnica de minimização de um problema. Devido ao grande número de variáveis envolvidas foram estabelecidas prioridades, transformadas imediatamente em metas, que, ao término deste, considera-se como cumpridas.

Anexos

Anexo 1 Programas-Fonte

Anexo 1.1 Cadastro.c

```
/*---------------------------------------------------------------------
CADASTRO.C
Responsavel pelo tratamento dos dados digitados no formulario de cadastro
de redirecionamento de URLs.
Ultima alteracao:

29/07/97 - 16:30 por Luis Fernando Garcia
* INCLUI AS ALTERACOES NAS PAGINAS *

Autores: Luis Fernando Fortes Garcia
         Tatiano Pianezzola
---------------------------------------------------------------------*/

#include <stdio.h>
#include <string.h>
#include <pwd.h>
#include <sys/types.h>
#include <unistd.h>
#include "util.c"

#define MAX_ENTRIES 10000

/* Definicao das estruturas de dados */

typedef struct {
   char *name;
   char *val;
} entry;

/*---------------------------------------------------------------
Nome: TELA_CIMA
Funcao: Montar a parte superior das paginas HTML.
      Utiliza o padrao do Instituto de Informatica
---------------------------------------------------------------*/

void tela_cima()
{
printf("Content-type:text/html%c%c",10,10);
printf("<HTML><TITLE>Instituto de Inform&aacute;tica - UFRGS (TESTE)</TITLE>");
printf("<table width=600 border=0>");
printf("<BODY BGCOLOR=#FFE7DE>");
printf("<BODY BACKGROUND=http://143.54.10.13:8080/fundo.gif>");
printf("<MAP NAME=\"cima\">");
printf("<AREA SHAPE=RECT COORDS=\"0,70,150,92\"HREF=http://www.inf.ufrgs.br/location/localizacao.html
ALT=\"Organiza&ccedil;&atilde;o\">");
printf("<AREA SHAPE=RECT COORDS=\"153,71,300,92\" HREF=http://www.inf.ufrgs.br/courses/cursos.html
ALT=\"Cursos\">");
printf("<AREA SHAPE=RECT COORDS=\"452,72,603,92\"
HREF=http://www.inf.ufrgs.br/activities/atividades.html ALT=\"Atividades\">");
printf("<AREA SHAPE=RECT COORDS=\"511,46,577,67\" HREF=http://www.inf.ufrgs.br/find.html
ALT=\"Procurar\">");
```

```
printf("<AREA SHAPE=RECT COORDS=\"456,46,502,69\" HREF=http://www.inf.ufrgs.br/mapa.html
ALT=\"Mapa\">");
printf("<AREA SHAPE=RECT COORDS=\"391,46,447,67\" HREF=http://www.inf.ufrgs.br/ ALT=\"Página inicial
do Instituto de Informática\">");
printf("<AREA SHAPE=RECT COORDS=\"306,72,444,94\"
HREF=http://www.inf.ufrgs.br/personal/lecturers/professor.html ALT=\"Professores\">");
printf("<AREA SHAPE=default HREF=http://143.54.10.13:8080/>");
printf("</MAP>");
printf("<IMG SRC=\"http://143.54.10.13:8080/cima.gif\" WIDTH=600 HEIGHT=97 border=0  USEMAP=\"#cima\"
alt=\"Instituto de Inform&aacute;tica (TESTE)\">");
}

/*--------------------------------------------------------------------
Nome: TELA_BAIXO
Funcao: Montar a parte inferior das paginas HTML.
        Utiliza o padrao do Instituto de Informatica
--------------------------------------------------------------------*/
void tela_baixo()
{
printf("<MAP NAME=\"baixo\">");
printf("<AREA SHAPE=RECT COORDS=\"0,4,99,24\" HREF=http://www.inf.ufrgs.br/biblioteca/biblio.htm
ALT=\"Biblioteca\">");
printf("<AREA SHAPE=RECT COORDS=\"102,5,200,24\" HREF=http://guarani.cos.ufrj.br/~sbc/english.htm>");
printf("<AREA SHAPE=RECT COORDS=\"202,3,299,25\" HREF=ftp://caracol.inf.ufrgs.br ALT=\"FTP\">");
printf("<AREA SHAPE=RECT COORDS=\"302,4,399,26\" HREF=http://www.ufrgs.br ALT=\"UFRGS\">");
printf("<AREA SHAPE=RECT COORDS=\"401,4,499,24\" HREF=http://www.inf.ufrgs.br/geral/estado.html
ALT=\"Porto Alegre\">");
printf("<AREA SHAPE=RECT COORDS=\"502,5,598,25\" HREF=http://www.inf.ufrgs.br/webmaster/about.html
ALT=\"Webmaster\">");
printf("<AREA SHAPE=default HREF=http://143.54.10.13:8080>");
printf("</MAP>");
printf("<IMG SRC=\"http://143.54.10.13:8080/baixo.gif\" WIDTH=600 HEIGHT=28 border=0
USEMAP=\"#baixo\" alt=\"Barra de Navega&ccedil;&atilde;o\">");
printf("</body></html>");
printf("</table>");
}

/*--------------------------------------------------------------------
Nome: CONF_SENHA
Funcao: Verificar se a senha digitada no formulario de cadastro
        confere com a armazenada no arquivo .passwd do UNIX.
--------------------------------------------------------------------*/

int conf_senha(char *user, char *pass)
{
        char salt[3];
        struct passwd *p;
        p = getpwnam( user );
        strncpy( salt, p->pw_passwd, 2 );
return (!strcmp( crypt( pass, salt ), p->pw_passwd ));
}

/*--------------------------------------------------------------------
Nome: GRAVA
Funcao: gravar no arquivo CADASTRO.DAT os dados digitados no
        formulario.
--------------------------------------------------------------------*/

int grava(entry *entries)
{
                FILE *in, *out;
                char filename[256], file_temp[256];
                char *line,*aux;
                char comando[255];

    /* Reserva area de memoria */
```

```
        line = (char *) malloc(200);
        aux = (char *) malloc(200);

/* Abre os arquivos de dados */

sprintf(filename,"./../htdocs/cadastro.dat");
        in = fopen (filename, "r");

sprintf(file_temp,"./../htdocs/temp.dat");
        out = fopen (file_temp,"w");


        /* Caso haja problemas na abertura do arquivos exibe msg */

        if ( in == NULL )
 {
      tela_cima();
      printf("<H1> Serviccedil;o de REDIRECIONAMENTO de URLs");
      printf("<p>");
      printf("Desculpe, o arquivo do Banco de Dados natilde;o existe!</h1>");
      printf("<p>");
      printf("<p>");
      printf("<p>");
      tela_baixo();
            exit (1);
        }

        if ( out == NULL ){
      tela_cima();
      printf("<H1> Serviccedil;o de REDIRECIONAMENTO de URLs");
      printf("<p>");
      printf("Desculpe, o arquivo do Banco de Dados natilde;o existe!</h1>");
      printf("<p>");
      printf("<p>");
      printf("<p>");
      tela_baixo();
      exit (1);
        }

        /* Percorre os arquivos */

        while(fgets(line,200,in)){
                if(strncmp(line,entries[0].val,strlen(entries[0].val))){
                        /*printf("<br>%s",line);*/
                        fprintf(out,line);
                }
        }

        /* O caracter \t e utilizado como separador das urls */

        fprintf(out,entries[0].val);
        fprintf(out,"\t");
        fprintf(out,entries[1].val);
        fprintf(out,"\n");

/* Operacoes com os arquivos de dados */

        fclose(in);
        fclose(out);
in = fopen (filename, "w");
        out = fopen (file_temp, "r");

        if ( in == NULL ){
      tela_cima();
      printf("<H1> Serviccedil;o de REDIRECIONAMENTO de URLs");
```

```
            printf("<p>");
            printf("Desculpe, o arquivo do Banco de Dados natilde;o existe!</h1>");
            printf("<p>");
            printf("<p>");
            printf("<p>");
            tela_baixo();
                  exit (1);
              }

              if ( out == NULL ){
            tela_cima();
            printf("<H1> Serviccedil;o de REDIRECIONAMENTO de URLs");
            printf("<p>");
            printf("Desculpe, o arquivo do Banco de Dados natilde;o existe!</h1>");
            printf("<p>");
            printf("<p>");
            printf("<p>");
            tela_baixo();
            exit (1);
              }

              while(fgets(line,200,out)){
                        fprintf(in,line);
              }

              fclose(in);
              fclose(out);
              return (0);
}

/* ---------------------------------------------------------------
Nome: MAIN
Funcao: Funcao principal
--------------------------------------------------------------- */
main(int argc, char *argv[])
{
   entry entries[MAX_ENTRIES];
   register int x, m=0;
   int cl;
   char url[256];
   char *user, *tmp, *tmp2;
   int boolean=0;

   cl = atoi(getenv("CONTENT_LENGTH"));
   printf("Content-type: text/html%c%c",10,10);

   for (x=0; cl && (!feof(stdin)); x++)
   {
      entries[x].val = fmakeword(stdin,'&',&cl);
      plustospace(entries[x].val);
      unescape_url(entries[x].val);
      entries[x].name = makeword(entries[x].val,'=');
      m=x;
   }

   tmp = (char *) malloc(strlen(entries[0].val));
   user = (char *) malloc(10);
   tmp2 = user;

   strcpy(tmp,entries[0].val);

   if(strchr(tmp,'~') != NULL){
     while (*tmp != '~') tmp++;
     tmp++;

     while (*tmp != '/') *tmp2++ = *tmp++;
```

```
    boolean =1;
   }

   if (boolean){
            if(conf_senha(user, entries[2].val)){
              grava(entries);
      tela_cima();
      printf("<IMG SRC=/"barra.gif/" BORDER=0>");
              printf("<TABLE BORDER=0 WIDTH=600><TR><TD align=center><BR>");
              printf("<P><BR>");
              printf("<IMG SRC="sucesso.gif" BORDER=0>");
      printf("<BR><BR>");
      printf("</TD></TR></TABLE><P>");
      tela_baixo();
      }
      else
      tela_cima();
              printf("<IMG SRC=\"barra.gif\" BORDER=0>");
      printf("<TABLE BORDER=0 WIDTH=600><TR><TD align=center><BR>");
      printf("<P><P><BR>");
      printf("<IMG SRC="senha.gif" BORDER=0>");
      printf("<P><P><BR>");
      printf("</TD></TR></TABLE><P>");
      tela_baixo();
   }
   else{
    grava(entries);
    tela_cima();
    printf("<IMG SRC=/"barra.gif/" BORDER=0>");
    printf("<TABLE BORDER=0 WIDTH=600><TR><TD align=center><BR>");
    printf("<P><BR>");
    printf("<IMG SRC="sucesso.gif" BORDER=0>");
    printf("<BR><BR>");
    printf("</TD></TR></TABLE><P>");
    tela_baixo();
   }
}
```

Anexo 1.2 Trata404.c

```
/*-------------------------------------------------------------------------
TRATA404.C
Responsavel pelo redirecionamento - Executado pelo servidor HTTPd

Ultima alteracao:

29/07/97 - 16:30 por Luis Fernando Garcia
* INCLUI AS ALTERACOES NAS PAGINAS *

Autores: Luis Fernando Fortes Garcia
        Tatiano Pianezzola
-------------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include <pwd.h>
#include <sys/types.h>
#include <sys/wait.h>

/* Define o nome e o caminhamento do arquivo de dados */

#define ARQ_CADASTRO "/home/coruja/lgarcia/httpd/htdocs/cadastro.dat"
#define TAM_LINE 200

/* Define as estruturas de dados */
/* Pega as variaveis do formulario (nome e valor) */

typedef struct {
   char *name;
   char *val;
} entry;

/*-------------------------------------------------------------------
Nome: TELA_CIMA
Funcao: Montar a parte superior das paginas HTML.
        Utiliza o padrao do Instituto de Informatica
-------------------------------------------------------------------*/

void tela_cima()
{
printf("Content-type:text/html%c%c",10,10);
printf("<HTML><TITLE>Instituto de Inform&aacute;tica - UFRGS (TESTE)</TITLE>");
printf("<table width=600 border=0>");
printf("<BODY BGCOLOR=#FFE7DE>");
printf("<BODY BACKGROUND=http://143.54.10.13:8080/fundo.gif>");
printf("<MAP NAME=\"cima\">");
printf("<AREA SHAPE=RECT COORDS=\"0,70,150,92\"HREF=http://www.inf.ufrgs.br/location/localizacao.html
ALT=\"Organiza&ccedil;&atilde;o\">");
printf("<AREA SHAPE=RECT COORDS=\"153,71,300,92\" HREF=http://www.inf.ufrgs.br/courses/cursos.html
ALT=\"Cursos\">");
printf("<AREA SHAPE=RECT COORDS=\"452,72,603,92\"
HREF=http://www.inf.ufrgs.br/activities/atividades.html ALT=\"Atividades\">");
printf("<AREA SHAPE=RECT COORDS=\"511,46,577,67\" HREF=http://www.inf.ufrgs.br/find.html
ALT=\"Procurar\">");
printf("<AREA SHAPE=RECT COORDS=\"456,46,502,69\" HREF=http://www.inf.ufrgs.br/mapa.html
ALT=\"Mapa\">");
printf("<AREA SHAPE=RECT COORDS=\"391,46,447,67\" HREF=http://www.inf.ufrgs.br/ ALT=\"Página inicial
do Instituto de Informática\">");
printf("<AREA SHAPE=RECT COORDS=\"306,72,444,94\"
HREF=http://www.inf.ufrgs.br/personal/lecturers/professor.html ALT=\"Professores\">");
```

```
printf("<AREA SHAPE=default HREF=http://143.54.10.13:8080/>");
printf("</MAP>");
printf("<IMG SRC=\"http://143.54.10.13:8080/cima.gif\" WIDTH=600 HEIGHT=97 border=0  USEMAP=\"#cima\"
alt=\"Instituto de Inform&aacute;tica (TESTE)\">");
}

/*---------------------------------------------------------------------
Nome: TELA_BAIXO
Funcao: Montar a parte inferior das paginas HTML.
        Utiliza o padrao do Instituto de Informatica
---------------------------------------------------------------------*/
void tela_baixo()
{
printf("<MAP NAME=\"baixo\">");
printf("<AREA SHAPE=RECT COORDS=\"0,4,99,24\" HREF=http://www.inf.ufrgs.br/biblioteca/biblio.htm
ALT=\"Biblioteca\">");
printf("<AREA SHAPE=RECT COORDS=\"102,5,200,24\" HREF=http://guarani.cos.ufrj.br/~sbc/english.htm>");
printf("<AREA SHAPE=RECT COORDS=\"202,3,299,25\" HREF=ftp://caracol.inf.ufrgs.br ALT=\"FTP\">");
printf("<AREA SHAPE=RECT COORDS=\"302,4,399,26\" HREF=http://www.ufrgs.br ALT=\"UFRGS\">");
printf("<AREA SHAPE=RECT COORDS=\"401,4,499,24\" HREF=http://www.inf.ufrgs.br/geral/estado.html
ALT=\"Porto Alegre\">");
printf("<AREA SHAPE=RECT COORDS=\"502,5,598,25\" HREF=http://www.inf.ufrgs.br/webmaster/about.html
ALT=\"Webmaster\">");
printf("<AREA SHAPE=default HREF=http://143.54.10.13:8080>");
printf("</MAP>");
printf("<IMG SRC=\"http://143.54.10.13:8080/baixo.gif\" WIDTH=600 HEIGHT=28 border=0
USEMAP=\"#baixo\" alt=\"Barra de Navega&ccedil;&atilde;o\">");
printf("</body></html>");
printf("</table>");
}

/* ---------------------------------------------------------------
Nome: SEPARA_URLS
Funcao: Separa, a partir do arquivo de dados as urls nova e antiga
        url velha - aparece em primeiro na linha
        url nova - aparece em segundo na linha
        Separador = caracter \t = TAB
--------------------------------------------------------------- */
void separa_urls(char *line,char *old, char *new){

    strcpy(old,"");
    strcpy(new,"");

    while ( *line != '\t'){
            *old++ = *line++;
    }

   *old = '\0';
   line++;
   while ( *line != '\n'){
    *new++ = *line++;
    *new = '\0';
   }
   line-=strlen(line)+1;
}

/* ---------------------------------------------------------------
Nome: NEW
Funcao: Pesquisa no arquivo de dados
--------------------------------------------------------------- */
char *new(char *old){
   char *url_old, *url_new;    /* estas sao aquelas do cadastro.dat */
   char line[TAM_LINE];
   FILE *fp;

    /* Abre o arquivo cadastro.dat */
```

```
   if ( (fp = fopen(ARQ_CADASTRO, "r")) == NULL ){

   /* Erro na Abertura do arquivo */

     tela_acima();
            printf("<H3>Transa&ccedil;&atilde;o n&atilde;o efetuada!</H3>\n");
            printf("<H3>Motivo: Problema durante a abertura do banco de dados.</H3>\n");
            exit(1);
     tela_baixo();
   }

   /* Aloca memoria */

   url_old = (char *)malloc ( TAM_LINE / 2 );
   url_new = (char *)malloc ( TAM_LINE / 2 );

   /* Le linha-a-linha o arquivo de dados */

   while ((fgets (line, TAM_LINE, fp)!=NULL)){
            strcpy(url_old,"");
            strcpy(url_new,"");
            separa_urls(line,url_old,url_new);
            strcpy(line,"");

            /* Compara a url cadastrada com a solicitada
              Se achar entao retorna a url nova */

            if (strcmp(url_old, old))
                     continue;
            else
                     return url_new;
   }


   /* caso nao encontra a URL no BD.
      emite mensagem e leva o usuario para a pagina de busca*/

tela_acima();
printf("<IMG SRC=\"barra.gif\" BORDER=0>");
printf("<TABLE BORDER=0 WIDTH=600><TR><TD align=center><BR>");
printf("<P><P><BR>");
printf(" ");
printf("<table border=0 WIDTH=450><tr><td>");
printf("<IMG SRC="interrog.gif" BORDER=0></td><td align=center>");
printf("<IMG SRC="nencont.gif" BORDER=0><BR>");
printf("%s", old);
printf("</td></tr><tr><td></td><td align=center><BR>");
printf("<A HREF="http://www.inf.ufrgs.br/find.html"><IMG SRC="bot.gif" BORDER=0></A>");
printf("</TD></TR></TABLE>");
printf("</TD></TR></TABLE><P>");
tela_baixo();
exit(0);
}

/* ---------------------------------------------------------------
Nome: MAIN
Funcao: Funcao principal
--------------------------------------------------------------- */

void main(){

 char *redirect_url;
 char nova_url[100];
 char *old_url;
```

```
  /* Monta a URL no formato padrao */

  strcpy(nova_url,"");
  strcat(nova_url, "http://");
  strcat(nova_url, (char *)getenv("SERVER_NAME"));
  strcat(nova_url, ":");
  strcat(nova_url, (char *)getenv("SERVER_PORT"));
  strcat(nova_url, (char *)getenv("REDIRECT_URL"));

  /* Efetua a busca no banco de dados */

  old_url = new(nova_url);

  if (!strncmp(old_url,"http://143.54.10.13:8080",24)){
    /* Redireciona para a nova pagina - mesmo servidor */
          old_url+=24;
          printf("Location: %s\n\n",old_url);
  }
  else{

    /* Redireciona para a nova pagina - outro servidor */
    printf("Status: 302 Redirect\r\n");
          printf("Location: %s\n\n",old_url);
 }
}
```

Anexo 1.3. Util.C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>


#define LF 10
#define CR 13

void getword(char *word, char *line, char stop) {
   int x = 0,y;

   for(x=0;((line[x]) && (line[x] != stop));x++)
      word[x] = line[x];

   word[x] = '\0';
   if(line[x]) ++x;
   y=0;

   while(line[y++] = line[x++]);
}

char *makeword(char *line, char stop) {
   int x = 0,y;
   char *word = (char *) malloc(sizeof(char) * (strlen(line) + 1));

   for(x=0;((line[x]) && (line[x] != stop));x++)
      word[x] = line[x];

   word[x] = '\0';
   if(line[x]) ++x;
   y=0;

   while(line[y++] = line[x++]);
   return word;
}

char *fmakeword(FILE *f, char stop, int *cl) {
   int wsize;
   char *word;
   int ll;

   wsize = 102400;
   ll=0;
   word = (char *) malloc(sizeof(char) * (wsize + 1));

   while(1) {
      word[ll] = (char)fgetc(f);
      if(ll==wsize) {
         word[ll+1] = '\0';
         wsize+=102400;
         word = (char *)realloc(word,sizeof(char)*(wsize+1));
      }
      --(*cl);
      if((word[ll] == stop) || (feof(f)) || (!(*cl))) {
         if(word[ll] != stop) ll++;
         word[ll] = '\0';
               word = (char *) realloc(word, ll+1);
         return word;
      }
      ++ll;
   }
}
```

```
char x2c(char *what) {
   register char digit;

   digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A')+10 : (what[0] - '0'));
   digit *= 16;
   digit += (what[1] >= 'A' ? ((what[1] & 0xdf) - 'A')+10 : (what[1] - '0'));
   return(digit);
}

void unescape_url(char *url) {
   register int x,y;

   for(x=0,y=0;url[y];++x,++y) {
      if((url[x] = url[y]) == '%') {
         url[x] = x2c(&url[y+1]);
         y+=2;
      }
   }
   url[x] = '\0';
}

void plustospace(char *str) {
   register int x;

   for(x=0;str[x];x++) if(str[x] == '+') str[x] = ' ';
}

int rind(char *s, char c) {
   register int x;
   for(x=strlen(s) - 1;x != -1; x--)
      if(s[x] == c) return x;
   return -1;
}

int getline(char *s, int n, FILE *f) {
   register int i=0;

   while(1) {
      s[i] = (char)fgetc(f);

      if(s[i] == CR)
         s[i] = fgetc(f);

      if((s[i] == 0x4) || (s[i] == LF) || (i == (n-1))) {
         s[i] = '\0';
         return (feof(f) ? 1 : 0);
      }
      ++i;
   }
}

void send_fd(FILE *f, FILE *fd)
{
   int num_chars=0;
   char c;

   while (1) {
      c = fgetc(f);
      if(feof(f))
         return;
      fputc(c,fd);
   }
}

int ind(char *s, char c) {
```

```
    register int x;

    for(x=0;s[x];x++)
        if(s[x] == c) return x;

    return -1;
}

void escape_shell_cmd(char *cmd) {
    register int x,y,l;

    l=strlen(cmd);
    for(x=0;cmd[x];x++) {
        if(ind("&;`\"|*?~<>^()[]{}$\\\x0A",cmd[x]) != -1){
            for(y=l+1;y>x;y--)
                cmd[y] = cmd[y-1];
            l++; /* length has been increased */
            cmd[x] = '\\';
            x++; /* skip the character */
        }
    }
}
```

Anexo 2 Arquivos de Configuração do Servidor

Anexo 2.1 SRM.conf

```
#==========================================================================
# NCSA HTTPd (comments, questions to httpd@ncsa.uiuc.edu)
#==========================================================================
# This is the server resource configuration file.  With this document,
# you define the name space that users of your server see.
# See URL http://hoohoo.ncsa.uiuc.edu/ for HTTPd Documentation.
# Information specific to this file can be found at
# http://hoohoo.ncsa.uiuc.edu/docs/setup/srm/Overview.html
# Do NOT simply read the instructions in here without understanding
# what they do.  If you are unsure, consult the online docs.  You have been
# warned.
#==========================================================================


#==========================================================================
# Name Space Options
#--------------------------------------------------------------------------
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.

DocumentRoot /home/coruja/lgarcia/httpd/htdocs

# UserDir: The name of the directory which is appended onto a user's home
# directory if a ~user request is recieved.

UserDir public_html

# Redirect allows you to tell clients about documents which used to exist in
# your server's namespace, but do not anymore. This allows you to tell the
# clients where to look for the relocated document.
# Format: Redirect fakename url

Redirect /HTTPd/ http://hoohoo.ncsa.uiuc.edu/

# Aliases: Add here as many aliases as you need. The format is
# Alias fakename realname

Alias /icons/ /home/coruja/lgarcia/httpd /icons/

# ScriptAlias: This controls which directories contain server scripts.
# Format: ScriptAlias fakename realname

ScriptAlias /cgi-bin/ /home/coruja/lgarcia/httpd/cgi-bin/

#==========================================================================
# Directory Indexing
#--------------------------------------------------------------------------
# If a user requests a document (URL) from your server ending in /, the
# server will attempt to "index" the directory.  It will first look for
# a file matching the DirectoryIndex directive in order.  If no files
# exist, and Indexing is allowed for that directory, the server will provide
# an index that it generates itself.  These options allow you to modify the
# look of that index.

# DirectoryIndex: Name of the file to use as a pre-written HTML
# directory index.  These files are used if a directory is referenced.
```

```
DirectoryIndex index.html index.shtml index.cgi

# IndexOptions

IndexOptions FancyIndexing

# AddIcon tells the server which icon to show for different files or filename
# extensions

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image.gif) image/*
AddIconByType (SND,/icons/sound.gif) audio/*
AddIcon /icons/movie.gif .mpg .qt
AddIcon /icons/binary.gif .bin

AddIcon /icons/back.xbm ..
AddIcon /icons/menu.gif ^^DIRECTORY^^
AddIcon /icons/blank.xbm ^^BLANKICON^^

# DefaultIcon is which icon to show for files which do not have an icon
# explicitly set.

DefaultIcon /icons/unknown.xbm

# AddDescription allows you to place a short description after a file in
# server-generated indexes.
# Format: AddDescription "description" filename

# ReadmeName is the name of the README file the server will look for by
# default. Format: ReadmeName name
#
# The server will first look for name.html, include it if found, and it will
# then look for name and include it as plaintext if found.
#
# HeaderName is the name of a file which should be prepended to
# directory indexes.

ReadmeName README
HeaderName HEADER

# IndexIgnore is a set of filenames which directory indexing should ignore
# This doesn't use full regexp syntax, perhaps it should . . .
# Format: IndexIgnore name1 name2...

IndexIgnore */.??* *~ *# */HEADER* */README*

#==========================================================================
# Content Type and Mime Configuration
#--------------------------------------------------------------------------
# Although NCSA HTTPd doesn't fully support the content-negotiation that
# exists in HTTP/1.1, it does attempt to correctly identify different
# encodings and types of files it serves.  The following options specify
# how it does this

# DefaultType is the default MIME type for documents which the server
# cannot find the type of from filename extensions.

DefaultType text/plain

# AddType allows you to tweak mime.types without actually editing it, or to
# make certain files to be certain types.
# Format: AddType type/subtype ext1

# AddEncoding allows you to have certain browsers (Mosaic/X 2.1+) uncompress
# information on the fly. Note: Not all browsers support this.
```

```
#AddEncoding x-compress Z
#AddEncoding x-gzip gz

# The following are known to the server as "Magic Mime Types"  They allow
# you to change how the server perceives a document by the extension
# The server currently recognizes the following mime types for server side
# includes, internal imagemap, and CGI anywhere.  Uncomment them to use them.
# Note: If you disallow (in access.conf) Options Includes ExecCGI, and you
# uncomment the following, the files will be passed with the magic mime type
# as the content type, which causes most browsers to attempt to save the
# file to disk.

#AddType text/x-server-parsed-html .shtml
#AddType text/x-imagemap .map
#AddType application/x-httpd-cgi .cgi

#============================================================================
# Misc Server Resources
#----------------------------------------------------------------------------
# AccessFileName: The name of the file to look for in each directory
# for access control information and directory specific configuration

AccessFileName .htaccess

# If you want to have files/scripts sent instead of the built-in version
# in case of errors, uncomment the following lines and set them as you
# will.  Note: scripts must be able to be run as if the were called
# directly (in ScriptAlias directory, for instance)

# 302 - REDIRECT
# 400 - BAD_REQUEST
# 401 - AUTH_REQUIRED
# 403 - FORBIDDEN
# 404 - NOT_FOUND
# 500 - SERVER_ERROR
# 501 - NOT_IMPLEMENTED

#ErrorDocument 302 /cgi-bin/redirect.cgi
#ErrorDocument 500 /errors/server.html
#ErrorDocument 403 /errors/forbidden.html

ErrorDocument 404 /cgi-bin/trata404.cgi
```

Anexo 3 Formulários HTML

Anexo 3.1 Cadastro.HTML

```
<HTML><TITLE>Formul&aacute;rio de Cadastro de URLs</TITLE>

<BODY BGCOLOR=#FFE7DE>
<BODY BACKGROUND=fundo.gif>
<MAP NAME="cima">

<AREA SHAPE=RECT COORDS="0,70,150,92" HREF=http://www.inf.ufrgs.br/location/localizacao.html
ALT="Organiza&ccedil;&atilde;o">
<AREA SHAPE=RECT COORDS="153,71,300,92" HREF=http://www.inf.ufrgs.br/courses/cursos.html
ALT="Cursos">
<AREA SHAPE=RECT COORDS="452,72,603,92" HREF=http://www.inf.ufrgs.br/activities/atividades.html
ALT="Atividades">
<AREA SHAPE=RECT COORDS="511,46,577,67" HREF=http://www.inf.ufrgs.br/find.html ALT="Procurar">
<AREA SHAPE=RECT COORDS="456,46,502,69" HREF=http://www.inf.ufrgs.br/mapa.html ALT="Mapa">
<AREA SHAPE=RECT COORDS="391,46,447,67" HREF=http://www.inf.ufrgs.br/ ALT="Página inicial do
Instituto de Informática">
<AREA SHAPE=RECT COORDS="306,72,444,94"
HREF=http://www.inf.ufrgs.br/personal/lecturers/professor.html ALT="Professores">
<AREA SHAPE=default HREF=http://143.54.10.13:8080/>
</MAP>
<IMG SRC="cima.gif" WIDTH=600 HEIGHT=97 border=0  USEMAP="#cima" alt="Instituto de
Inform&aacute;tica (TESTE)">
<IMG SRC="barra.gif" BORDER=0><p>
<TABLE BORDER=0 WIDTH=600><TR><TD>
<form method=POST action="/cgi-bin/cadastro.cgi">
<B>Preencha o formulário abaixo:</B>
<HR>
<table border=0><tr><td>
<B>URL ANTIGA:</B></td><td><input  size=50 maxlength=50 name="url_old" value="">
</td></tr><tr><td>
<B>URL NOVA:</B></td><td><input size=50 maxlength=50 name="url_new" value="">
</td></tr></table>
<hr>

<FONT COLOR="Red"><B>ATEN&Ccedil;&Atilde;O:</B></FONT>
Para páginas pessoais ( no formato http://143.54.10.13:8080/~user/pag.html )
&eacute; necess&aacute;rio digitar a senha do usu&aacute;rio.
<p>
<B>PASSWORD: </B><input size=8 maxlength=8 type=password name="passwd" value="">
<HR>
<p>
<H2>
<p align=center><input type="submit" value="Dispara Procedimento">
        <input type="reset" value="Cancela Procedimento">
</H2></form>


</TD></TR></TABLE>

<MAP NAME="baixo">

<AREA SHAPE=RECT COORDS="0,4,99,24" HREF=http://www.inf.ufrgs.br/biblioteca/biblio.htm
ALT="Biblioteca">
<AREA SHAPE=RECT COORDS="102,5,200,24" HREF=http://guarani.cos.ufrj.br/~sbc/english.htm>
<AREA SHAPE=RECT COORDS="202,3,299,25" HREF=ftp://caracol.inf.ufrgs.br ALT="FTP">
<AREA SHAPE=RECT COORDS="302,4,399,26" HREF=http://www.ufrgs.br ALT="UFRGS">
<AREA SHAPE=RECT COORDS="401,4,499,24" HREF=http://www.inf.ufrgs.br/geral/estado.html ALT="Porto
Alegre">
<AREA SHAPE=RECT COORDS="502,5,598,25" HREF=http://www.inf.ufrgs.br/webmaster/about.html
ALT="Webmaster">
```

```
<AREA SHAPE=default HREF=http://143.54.10.13:8080>
</MAP>
<IMG SRC="baixo.gif" WIDTH=600 HEIGHT=28 border=0 USEMAP="#baixo" alt="Barra de
Navega&ccedil;&atilde;o" >

</body></html>
```

Bibliografia

[ACR 97] ACRESCENTANDO conteúdo dinâmico para vários browsers e versões. [S.l]: Netscape World, 1997.

[AKS 88] AKSCYN, Robert; MCCRACKEN, Donald; YOSER, Elise. KMS: A Distributed Hypermedia System for Managing Knowledge In Organizations. **Communications of the ACM,** New York, v. 31. n. 7. p. 820- 835, July 1988.

[AND 94] ANDREWS, Keith; KAPPE, F.; MAURER, Hermann. The Hyper-G Network Information System. **J.UCS**, [S.l], v.1, n. 4, Apr. 1995.

[BER 92] BERNERS-LEE, Tim; CAILLIAU, Robert. World-Wide Web. **Computing in High Energy Phisics**, Annecy, France, 1992.

[BRA 97] BRA, P. M. E. de. **Hypermedia Structures and Systems**. Disponível por WWW em http://wwwis.win.tue.nl/~debra/cursus/static/distribution. html. (1997).

[BUR 91] BURGER, Andrew et al. The Virtual Notebook System. In: ACM CONFERENCE ON HYPERTEXT, 3., 1991, Santo Antonio, Texas. **Proceedings ...** New York: ACM, c1991.

[CAE 95] COMPUTER Aided Engeneering Network, University of Michigan. **ChURL "Check URL" (Perl4 Script)**. Disponível por WWW em http://www-personal.engin.umich.edu/~yunke/scripts/churl/. (1995).

[CAR 94] CARR, Leslie et al. The Microcosm Link Service and its application to the World-wide Web. In: WWW CONFERENCE, 1., 1995, Geneva. **Proceedings ...** Geneva: [s.n]. 1995.

[CAR 9?] CARR, Leslie et al. **The Distributed Link Service: A Tool for Publisher, Author and Readers**. [S.l]: Multimedia Research Group, Departament of Electronics & Computer Science, University of Southampton. 199?.

[CER 96] CERN. **Customizing Error Messages**. Disponível por WWW em http://www.w3.org/pub/WWW/Daemon/User/Error.html. (1996).

[CON 87] CONKLIN, Jeff. Hypertext: An Introduction and Survey. **Computer**, New York, p. 17, Sept. 1987.

[DER 95] DEROURE, David. **Networked Multimedia:** The Need for Hypermedia Link Services.. Disponível por WWW em http://vim.esc.soton.ac.uk/ telecom.html. (1995).

[DIS 95] DISTRIBUTED Link Service. Disponível por WWW em http://wwwcosm.ecs.soton.ac.uk/dls/dls.html. (1995).

[DIS 96] DISTRIBUTION and Concurrency**. Hypermedia Structures and Systems.** Disponível por WWW em http://www.win.tue.nl/win/cs /is/debre/cursus/. (1996).

[DUC 94] DUCHIER, Denys. **State of the Art Review on Hypermedia Issues And Applications**. Disponível por WWW em http://www.isg.sfu.ca/~duchier/misc/ hypertext_review/index.html. (1994).

[FIE 95] FIELDING, Roy**.** Maintaining Distributed Hypertext Infostructures: Welcome to MOMspider's Web. In: INTERNATIONAL WORLD-WIDE WEB CONFERENCE (WWW94), 1., 1994, Geneva. **Proceedings ...** [S.l.:s.n.], 1994.

[GAR 96] GARDNER, Elizabeth. Dead-Links: Problema de proporções epidêmicas? **Web Week**, [S.l.], v.1, n.3, p.24, 1996.

[GRA 95] GRAAFF, Hans. **Checkbot.** Disponível por WWW em http://dutifp.twi.tudelft.nl: 8000/checkbot-info.html. (1995).

[HIL 94] HILL, Gary. HALL, Wendy. Extending the Microcosm Model to a Distributed Environment. In: ECHT'94. **Proceedings ...** [S.l.:s.n.], c1994.

[HIL 95] HILL, Gary et al. **Applying Open Hypertext Principles to the WWW**. Disponível por WWW em http://wwwcosm.esc.soton.ac.uk/Microcosm/conference95/mp.abstract.html. (1995).

[HYP 9?] HYPERTED - The Hypertext Editor for the Macintosh. Disponível por WWW em http://cmi.med.monash.edu.au/hyperted.htm. (199?).

[KAP 91] KAPPE, Frank; PANI, Gerald. **The Architeture of a A Massively Distributed Hypermedia System**. Disponível por WWW em ftp://ftp.unicamp.br/pub/Hyper-G/papers/Report341.ps.gz. (1991).

[KAP 93] KAPPE, Frank. Hyper-G, A Distributed Hypermedia System. INET'93, 1993, San Francisco. **Proceedings ...** [S.l.:s.n], 1993.

[KAP 95] KAPPE, Frank. **A Scalable Architeture for Maintaining Referential Integrity in Distributed Information Systems**. Disponível por WWW em ftp://ftp.unicamp.br/pub/Hyper-G/papers/p-flood.ps.Z. (1995).

[KNO 94] KNOWLES, David. **A distributed project documentation manager for HTML documents and the World-Wide Web**. Final Year Computer Systems. Limerick: University of Limerick, 1994.

[KOS 94] KOSTER, Martijn. **A Standard for Robot Exclusion**. Disponível por WWW em http://web.nexor.co.uk/mak/doc/robots/norobots.html. (1994).

[LAN 92] LANGE, Danny; OSTERBYE, Kasper; SCHÜTT, Helge. **Hypermedia Storage.** Denmark: Institute for Electronic Systems, 1992. (R-92-2009).

[LIU 94] LIU, Cricket et al. **Managing Internet Information Services**. USA: O´Reilly & Associates, 1994.

[MAI 93] MAIOLI, Cesare; SOLA, Stefano; VITALI, Fabio. **Wide-Area Distribution Issues in Hypertext Systems**. Bologna: Laboratory for Computer Science, University of Bologna, 1993. (Technical Report UBLCS-93-24).

[MUL 97] MULTICOSM. **Microcosm: A Technical Overview**. Disponível por WWW em http://www.multicosm.com/flyer.html. (1997).

[NET 96] NETSCAPE. **Netscape Communications Server Installation and Reference Guide**. [S.l]: Netscape Press, 1996.

[NEW 95] NEWBERY, Michael. **Katipo - A Web Lurker**. Disponível por WWW em http:// www.wuw.ac.nz/~newbery/Katipo.html. (1995).

[NIE 90] NIELSEN, Jakob. **Hypertext and Hypermedia**. San Diego, USA: Academic Press Inc, 1990.

[PEA 89] AMY, Pearl. Sun's Link Service: A Protocol for Open Linking. In: HYPERTEXT, 1989. **Proceeding ...** [S.l:s.n.], 1989.

[PIT 95] PITKOW, James. **The html_analyser-0.30 README file**. Disponível por WWW em http://www.gatech.edu/pitkow/html_analyser/README.html. (1995).

[RAD 91] RADA, Roy. **From Text to Expertext**. [S.l..]: Mac-GrawHill Book Company, 1991.

[REG 94] REGENTS of the University of California. **MOMSpider - Distribution Information**. Disponível por WWW em http://www.ics.uci.edu/ WebSoft/MOMspider/. (1994).

[SHA 93] SHACKELFORD, Douglas. SMITH, John, SMITH, F. Donelson. The Architeture and Implementation of a Distributed Hypermedia Storage System. In: HYPERTEXT, 1993. **Proceedings ...** [S.l:s.n.], 1993.

[STA 92] STATUS codes in HTTP. Disponível por WWW em http://www.w3.org/ pub/WWW/Protocols/HTTP/HTRESP.html. (1992).

[STO 95] STOTTS, David. Timed Links Solve the "Stale URL" Problem. **SIGLINK Newsletter** [S.l], v. 4, n. 3, p. 6-7, Dec. 1995.

[SHI 89] SHIPMAN, Frank; CHANEY, Jesse; GORRY, Anthony. Distributed Hypertext for Collaborative Research: The Virtual Notebook System. In: HYPERTEXT, 1989. **Proceeding ...** [S.l:s.n.], 1989.

[TRE 95] TRELOAR, Andrew. **Scholarly Publishing and the Fluid World-Wide Web**. Disponível por WWW em http://www.csu.edu.au/special/ conference/apwww95/papers95/atreloar/atreloar.html. (1995).

[URL 94] URL-minder: Your Own Personal Web Robot! Disponível por WWW em http://www.netmind.com/URL-minder/URL-minder.html. (1994).

[VAN 94] VANZYL, Adrian; CESNIK, Branko. **Open Hypertext Systems**: An Examination of Requeriments, and Analysis of Implementation Strategies, comparing Microcosm, HyperTED, and the World-wide web. Australia: Monash University, 1994.

[VER 95] VERIFY Web Links. Disponível por WWW em http://wsk.eit.com/wsk/dist/doc/admin/webtest/verify_links.html. (1995).

[WHY 95] WHY is quality Assurance necessary?. Disponível por WWW em http://www.erin.gov.au/technical/management/quality_ass/quality_ass.html. (1995).

[WOO 91] WOODHEAD, Nigel. **Hypertext & Hypermedia** - Teory and Aplications. [S.l.]: Addison Wesley, 1991.

[XIE 95] XIE, George. **CHECKER - WWW Link Checker (UNIX version)**. Disponível por WWW em http://www.ugrad.cs.ubc.ca/spider/q7f192/branch/ checker.html. (1995).