LUÍS ARTHUR FERREIRA PINTO

AUTORIA GRÁFICA DE ESTRUTURAS DE DOCUMENTOS HIPERMÍDIA NO SISTEMA HYPERPROP

DISSERTAÇÃO DE MESTRADO

DEPARTAMENTO DE INFORMÁTICA

Rio de Janeiro, 30 de Agosto de 2000

LUÍS ARTHUR FERREIRA PINTO

AUTORIA GRÁFICA DE ESTRUTURAS DE DOCUMENTOS HIPERMÍDIA NO SISTEMA HYPERPROP

Dissertação de Mestrado apresentada ao Departamento de Informática da PUC/RJ, como parte dos requisitos para obtenção do título de Mestre em Informática: Ciência da Computação.

Orientador: Luiz Fernando Gomes Soares

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 30 de Agosto de 2000

Este trabalho é dedicado

aos meus pais.

AGRADECIMENTOS

Em primeiro lugar, ao meu orientador, professor Luiz Fernando Gomes Soares, pelos incentivos, críticas e discussões que tornaram possível a conclusão deste trabalho.

Aos colegas do Laboratório TeleMídia da PUC-Rio, pela amizade e ajuda indispensáveis, em especial a Débora Muchaluat e Rogério Rodrigues que colaboraram de forma fundamental para a realização desta tese.

Aos amigos Daniel Malaguti, Flávio Oliveira, Luiz Eduardo Matta, Tula Kraiser, entre outros, que acreditaram que este dia chegaria. Em especial, a Ram Rajagopal, pelo pontapé inicial, e a Janaína Tavares, pela compreensão dos momentos de que este trabalho nos privou.

A todos os professores e funcionários do Departamento de Informática da PUC-Rio pelos conhecimentos adquiridos e pela ajuda prestada ao longo do mestrado.

Agradeço também à Universidade de Auburn, pela liberação da utilização de sua biblioteca de desenho de grafos.

Por fim, à CAPES pelo suporte financeiro dado à pesquisa em que se baseou esta dissertação.

RESUMO

Um dos maiores desafios dos sistemas hipermídia atuais é oferecer uma interface gráfica que facilite a visualização da estruturas de documentos no espaço de informações. A grande quantidade de nós e elos presentes nos mapas estruturais contribui para o agravamento do conhecido problema de desorientação do usuário. Esta dissertação visa propor uma ferramenta gráfica para a navegação e autoria de estruturas de documentos hipermídia, compostos por contextos aninhados. A ferramenta, parte integrante do sistema HyperProp, possui como principal característica um mapa ativo que permite filtrar a informação apresentada, de modo que somente os nós relevantes para o usuário sejam mostrados. Além disso, técnicas de *layout* de grafos são utilizadas para melhorar a legibilidade do mapa. As funções de edição utilizam os mesmos recursos gráficos para dar suporte à autoria de todos os componentes do documento, definidos no modelo NCM (*Nested Context Model*).

Palavras-chave: sistema HyperProp, autoria hipermídia, olho-de-peixe, *layout* de grafos

SUMÁRIO

LISTA DE FIGURAS

Capítulo 1

Introdução

Os primeiros sistemas hipermídia eram aplicações fechadas, baseadas em soluções isoladas e em formatos proprietários. Em geral, estes sistemas não eram distribuídos, ficando restritos a operar em uma única estação. Todos eles eram construídos em torno de um tipo de banco de dados ou de um formato de arquivo próprio, o que dificultava ainda mais a distribuição e a integração com outros sistemas e aplicações.

A *World-Wide Web* [8], desenvolvida originalmente para facilitar a troca de dados entre pesquisadores da comunidade da física de altas energias, tornou-se o maior e mais bem sucedido sistema hipermídia distribuído atualmente em operação, devido principalmente à facilidade de seu uso e à utilização de protocolos de comunicação abertos e padronizados. A WWW foi um avanço considerável na abertura dos sistemas hipermídia, ao conseguir uma larga distribuição e o estabelecimento de padrões universalmente aceitos; contudo ela ainda apresenta várias limitações.

A mesma simplicidade do modelo hipertexto da WWW, que possibilitou sua enorme popularidade, impôs graves restrições na autoria e apresentação dos dados hipermídia. Entre

outros problemas, há o fato dos elos estarem inseridos nos próprios documentos e a falta de mecanismos de estruturação dos documentos. É comum, por exemplo, navegar pela *web* e se deparar com elos que não levam a nenhum documento (*dangling links*), porque este foi movido ou apagado. Além disso, não há como visualmente orientar o usuário de forma a responder questões simples como: "Para onde posso ir agora?" ou "Quais documentos apontam para cá?". Assim, o leitor de um hiperdocumento, em geral, não tem idéia da posição em que se encontra, a não ser que o autor explicitamente indique isto. O modelo simples em que se baseia a *web*, ao não permitir soluções limpas para os problemas citados, motiva o estudo e desenvolvimento de modelos hipermídia mais sofisticados, com maior poder de expressão.

Recentemente, com o desenvolvimento de aplicações multimídia distribuídas e da própria WWW, um outro ponto focal das pesquisas tem sido a busca de soluções para a integração entre os sistemas hipermídia e as demais aplicações. O objetivo é torná-los um componente *middleware*, totalmente inserido no ambiente do usuário, que forneça seus serviços de hipertexto a qualquer aplicação. Este esforço levou à definição e ao desenvolvimento dos chamados Sistemas Hipermídia Abertos (OHS), sistemas que permitem a adição de novas funcionalidades e provêem serviços hipermídia às aplicações clientes, através de uma arquitetura aberta e distribuída, sem limitações quanto ao suporte a diferentes mídias e formatos de dados [40].

No âmbito dos OHS, a proposta do Laboratório TeleMídia do Departamento de Informática da PUC-Rio se destaca na formalização de um modelo hipermídia de grande poder de expressão – o Modelo de Contextos Aninhados – e na criação de um protótipo para a autoria e consulta de documentos hipermídia – o sistema HyperProp –, que permite definir a estrutura hierárquica e o comportamento temporal dos documentos. O sistema HyperProp já

foi tema de vários trabalhos anteriores [12,32,44], que apresentaram soluções empregadas isoladamente nas primeiras versões do sistema. A principal motivação desta dissertação é reunir e aperfeiçoar algumas destas soluções, para compor a ferramenta de autoria estrutural de documentos hipermídia do sistema.

1.1 Objetivos

Uma das principais vantagens dos sistemas hipermídia é sua capacidade de organizar documentos de diferentes maneiras, criando uma forma de acesso não-linear à informação. Entretanto, em geral, estes sistemas produzem dados multimídia complexos e fartamente conectados, que dificultam a orientação do leitor ao navegar pelos documentos. Bons navegadores (*browsers*) são componentes fundamentais dos sistemas hipermídia, principalmente os sistemas de larga escala. É função desejável do navegador mostrar um mapa de todo o documento ou de parte dele, como forma de reduzir a desorientação do leitor, fornecendo uma importante medida de contexto e espaço. Além de manter a noção de localização, a visão geral dos documentos contribui para a diminuição da sobrecarga cognitiva dos leitores, que são obrigados a decidir dentre as várias opções de elos, quais devem seguir durante a navegação pelo hiperdocumento. É mais fácil reconhecer um padrão num mapa do que lembrar-se do caminho sozinho [10].

O presente trabalho tem como principal objetivo construir uma ferramenta gráfica para a autoria e apresentação de estruturas de documentos hipermídia, a ser utilizada no módulo cliente do sistema HyperProp. Esta ferramenta deve oferecer uma interface para a visualização da hierarquia e dos relacionamentos dos documentos, possibilitando a navegação por todo o espaço de informações hipermídia. O ambiente deverá ser constituído por

navegadores que apresentam uma visão global da estrutura dos documentos na forma de um mapa ativo, composto por um diagrama de nós e elos.

Apresentar o espaço de informações na forma de um grafo não é uma tarefa simples, pois este precisa ser desenhado de maneira clara para que auxilie de fato a navegação do usuário. Algoritmos para a disposição automática do desenho (*layout*) de grafos ajudam a evitar que dois nós se sobreponham, ou que haja vários cruzamentos de elos, etc. Não está no escopo desta dissertação aprofundar o estudo do desenho de grafos, um problema comprovadamente bastante complexo, e sim encontrar um algoritmo adequado para o *layout* de grafos compostos para que seja empregado no navegador.

A legibilidade do mapa estrutural dos documentos não é obtida somente com o desenho correto do grafo que o constitui. Geralmente este grafo é muito grande e possui uma alta conectividade, sendo imprescindível que o navegador possua algum mecanismo de filtragem que restrinja a informação exibida ao usuário. A solução encontrada para o navegador do HyperProp utiliza uma técnica de filtragem baseada na estratégia olho-de-peixe para grafos compostos [33]. Esta técnica permite que apenas os detalhes referentes à posição do usuário durante a navegação sejam exibidos. Nesta dissertação são propostas otimizações no algoritmo de cálculo da visão olho-de-peixe e incluídas novas funcionalidades, como o suporte a elos *n*:*m*, a escolha do grau de detalhamento dos mapas e a possibilidade de configuração da filtragem a ser realizada.

O ambiente de autoria do HyperProp desenvolvido nesta dissertação tira proveito da interface gráfica formada por um grafo composto para disponibilizar ao usuário uma maneira intuitiva e simples para a edição dos componentes de um documento, definidos no modelo hipermídia no qual o sistema é baseado. Através da seleção de nós no mapa, poderão ser facilmente criados composições, elos e âncoras em documentos.

1.2 Organização da Dissertação

O Capítulo 2 inicia-se com os conceitos básicos dos sistemas hipermídia e dos OHS, necessários ao entendimento do restante do trabalho. Neste capítulo são apresentados também alguns dos principais sistemas hipermídia, destacando as soluções encontradas para a apresentação do mapa de documentos e o suporte à autoria em seus navegadores. No Capítulo 3 é exposto o HyperProp, sistema para o qual foi desenvolvida a ferramenta gráfica. São analisados sua arquitetura, o modelo conceitual de dados (NCM) em que é baseado e os principais aspectos referentes a sua interface gráfica.

A solução encontrada para o desenho dos mapas de documentos hipermídia dos navegadores do HyperProp é abordada no Capítulo 4. É apresentado o algoritmo de *layout* automático utilizado e detalhadas as extensões desenvolvidas para adequá-lo às características dos grafos que representam as estruturas de documentos baseados no NCM.

No Capítulo 5, é apresentada a definição da visão olho-de-peixe e analisada sua aplicação em grafos compostos. As novas funcionalidades acrescentadas ao mecanismo de filtragem também são tratadas neste capítulo. O Capítulo 6 aborda as facilidades oferecidas pelo sistema na autoria de estruturas de documentos hipermídia, apresentando os detalhes referentes à implementação dos navegadores que compõem o HyperProp.

Finalmente, o Capítulo 7 é dedicado a uma avaliação geral desta dissertação, ressaltando suas contribuições e propondo algumas sugestões para trabalhos futuros. Neste capítulo, também são realizadas comparações entre as técnicas de desenho e filtragem de grafos estudadas, e entre os sistemas apresentados no Capítulo 2 e o HyperProp.

Capítulo 2

Trabalhos Relacionados

O hipertexto tem revolucionado a capacidade de criação e leitura, possibilitando ao leitor explorar diversas alternativas e assimilar o conhecimento de diferentes formas. A flexibilidade do acesso é baseada nos conceitos de navegação, anotação e apresentação diferenciada. Os sistemas hipermídia constituem a classe de sistemas de gerenciamento de informação que permitem a seus usuários criar, organizar e compartilhar informações apresentadas em diversas mídias, tendo como base o hipertexto.

Em geral, tais sistemas se utilizam de navegadores gráficos para a exibição da estrutura dos documentos hipermídia, auxiliando a navegação do usuário e oferecendo acesso à autoria destes documentos. A construção dos navegadores não é uma tarefa trivial: existem dificuldades, principalmente quanto à interface dos mecanismos de edição embutidos e à forma como o espaço de informações deve ser apresentado. Estudar os trabalhos relacionados nesta área é interessante para analisar quais seriam as melhores técnicas para a implementação destas ferramentas.

Este capítulo inicia-se com uma noção geral sobre os conceitos básicos do hipertexto. A definição dos atuais sistemas hipermídia abertos também é apresentada, assim como suas características e vantagens em relação aos primeiros sistemas. Em seguida, são descritos alguns exemplos de sistemas hipermídia, em especial aqueles que apresentam soluções relacionadas à interface gráfica dos navegadores de estrutura de documentos.

2.1 Introdução

A palavra *hipertexto* foi usada pela primeira vez por Ted Nelson em 1965, para denominar "um material escrito ou pictórico interconectado de uma maneira complexa, que não poderia ser convenientemente representado sobre o papel" [38]. Poderia conter sumários ou mapas de seu conteúdo e de seus inter-relacionamentos, assim como anotações, acréscimos ou notas de rodapé das pessoas que o examinassem. De acordo com Rada [41], o termo hipertexto está relacionado com a idéia do *espaço hiperbólico* do matemático F. Klein (1849-1929). Em seus trabalhos, Klein empregava o termo hiperespaço para descrever a geometria aplicada a várias dimensões. Hipertexto seria, portanto, uma extensão multi-dimensional de um texto, que potencialmente poderia modelar o hiperespaço de conceitos que estaria contido no texto.

Hipertexto também pode ser definido como uma abordagem não-linear para o gerenciamento de informação, na qual os dados são armazenados em uma rede de nós conectados por elos. Os nós geralmente representam uma única idéia ou conceito, e os elos indicam associações entre estas idéias. Com o hipertexto, o leitor pode seguir pelos elos entre os nós e obter novos dados, de uma maneira não-seqüencial. A estrutura do hipertexto equipara-se à percepção humana na organização das idéias. Nosso pensamento é

essencialmente organizado como uma rede semântica de conceitos, unidos uns com os outros por meio de associações [5].

A coleção destes nós relacionados entre si, formando um conjunto lógico de informação, é chamada de hiperdocumento (ou simplesmente, de documento). A Figura 2.1 mostra o diagrama de um hiperdocumento, com vários nós e elos interligando-os. Um navegador (*browser*) exibe uma representação do conteúdo do nó corrente, nó A no exemplo. Os elos são indicados no navegador por meio de suas âncoras, pequenas regiões dos nós, tipicamente palavras, botões ou imagens. Ativando-as, o usuário pode navegar através da informação, seguindo os elos definidos pelo autor.

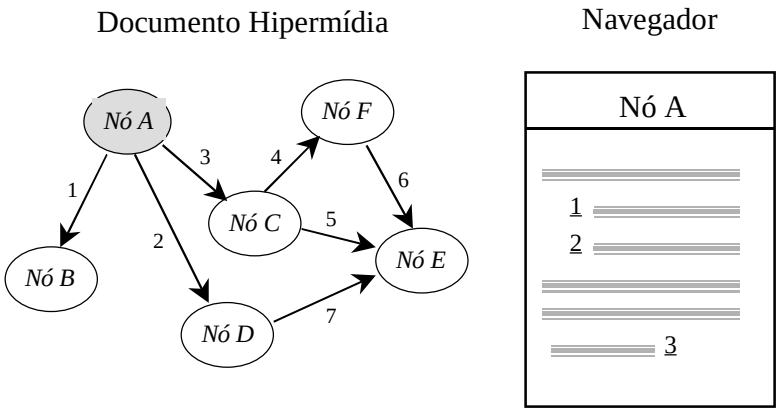


Figura 2.1 – Componentes de um sistema hipermídia simples.

Idealmente, os elos são bidirecionais, permitindo ao usuário percorrer o caminho inverso daquele definido pelo autor e assim descobrir quais documentos fazem referência a um determinado nó. Elos podem ser classificados, segundo Conklin, como referenciais ou hierárquicos [10]. Os elos referenciais são os mais comuns num hipertexto, representando as associações entre os conceitos, e os hierárquicos são usados para organizar os nós em uma estrutura predefinida, geralmente apresentando alguma semântica. Num hiperdocumento que representasse um livro, elos hierárquicos indicariam as subdivisões em capítulos e seções, enquanto que os elos referenciais apontariam para notas de rodapé, imagens ou outras

informações relacionadas. Nota-se assim que para preservar as duas semânticas de relações, são necessárias entidades distintas para representá-las.

A informação puramente escrita está se tornando cada vez mais rara hoje em dia. Os avanços técnicos dos equipamentos para armazenamento, transmissão e exibição permitiram uma divulgação mais rica da informação, combinando-se diversas mídias de um modo integrado. Portanto, é comum que atualmente em um documento hipertexto, os nós não se restrinjam somente a texto, podendo conter também imagens, áudio, vídeo, ou qualquer outra forma de dados, apresentando relacionamentos entre nós de mídias distintas. Neste caso, o termo mais apropriado para classificar este documento é hipermídia. Entretanto, é comum utilizar os termos hipertexto e hipermídia indistintamente.

As técnicas de hipermídia têm sido utilizadas para uma grande variedade de aplicações, incluindo livros eletrônicos em CD-ROMs, sistemas de ajuda *on-line*, sistemas de treinamento, assim como a *World Wide Web* e outros sistemas utilizados para pesquisar informações armazenadas de forma distribuída.

2.2 Sistemas Hipermídia Abertos

Os primeiros sistemas hipermídia eram aplicações auto-suficientes e fechadas, no qual o material hipermídia podia ser criado e consultado. Em geral, estes sistemas não eram distribuídos, ficando restritos a operar em uma única estação. Todos eles eram construídos em torno de um tipo de banco de dados ou de um formato de arquivo proprietário, o que impossibilitava a troca de informações e a compatibilidade com outros sistemas e aplicações. Para contornar estes problemas, que se acentuavam à medida que as pesquisas no projeto de

sistemas hipermídia progrediam, foi desenvolvida uma nova abordagem, introduzindo o conceito de sistema aberto.

Um sistema hipermídia aberto (*Open Hypertext System* - OHS) pode ser entendido como um componente intermediário, que oferece o serviço de hipertexto às aplicações existentes através de uma arquitetura aberta e possivelmente distribuída, formando um ambiente hipermídia único e global. Independente de seu sistema operacional e plataforma de *hardware*, uma aplicação qualquer pode usar as funcionalidades de um OHS e fornecer os serviços de hipertexto aos dados gerenciados por ela. O termo *aberto* significa que o sistema hipermídia especifica apenas um formato para a estrutura do modelo de dados, não havendo limitações quanto ao suporte a diferentes mídias e tipos de dados [40]. As características fundamentais de um OHS são sua modularidade e extensibilidade. Para atender a estas características, os sistemas precisam satisfazer certos requisitos [14,15], entre eles os citados a seguir.

Um sistema hipermídia aberto deve ser extensível quanto ao tamanho de sua base de dados e ao suporte a diferentes mídias e tipos de dados, sendo flexível o bastante para facilmente poder incorporar novos formatos. O OHS deve garantir a integridade dos elos e âncoras de um documento, quando este for editado ou movido. A criação de um elo ou de uma âncora num documento são processos que não podem modificar o conteúdo de seus dados. Isto é desejável por diversas razões: o arquivo que contém o documento pode estar armazenado em um servidor o qual o usuário não tenha permissão de escrita ou estar numa mídia própria apenas para leitura (como o CD-ROM). Além disso, porque a marcação das âncoras deve ser transparente para as aplicações que não dispõem do serviço de hipertexto, como, por exemplo, o processador de textos originalmente utilizado para criar o documento.

O desenvolvimento de sistemas abertos tem se mostrado vantajoso se comparado com os sistemas anteriores (fechados). Dentre os benefícios, há o intercâmbio de documentos entre vários sistemas, baseados em implementações e plataformas diferentes, permitindo uma rápida expansão da base de dados local, através de simples referências a informações externas, ou vice-versa. Outro benefício é a possibilidade de integração com outros sistemas de informação, particularmente aqueles que utilizam a Internet, como WWW, Gopher, FTP e WAIS. Ainda outro benefício dos OHS é a possibilidade de estender sua funcionalidade, via adição de novos módulos, e o suporte a diferentes formatos de dados.

Por fim, uma importante vantagem dos sistemas abertos é a sua interface integrada. Ao invés de oferecer um serviço hipertexto como uma aplicação isolada, o sistema estende esse serviço de uma maneira implícita a todo o ambiente do usuário. Assim, a navegação por hipertexto pode ser usada como um método genérico para a pesquisa e gerenciamento do espaço de informação do usuário.

No âmbito dos sistemas hipermídia abertos, destacam-se principalmente as propostas do Hyper-G, Madeus, Microcosm, DHM, Chimera e HyperProp, entre outros. Uma tendência que se observa é a preferência na integração destes sistemas com a WWW, valendo-se da ampla distribuição e dos padrões firmados por este sistema [1]. O desenvolvimento de OHS baseados em modelos hipertexto mais completos pretende eliminar as deficiências estabelecidas pela WWW e outros problemas tipicamente associados a sistemas distribuídos hipermídia de larga escala.

As seções restantes deste capítulo trazem uma breve introdução a alguns dos sistemas citados, destacando somente aqueles que apresentam soluções para a exibição de mapas estruturais de documentos e para o suporte à autoria em seus navegadores. O estudo de suas

interfaces gráficas é importante para uma melhor comparação com o trabalho desenvolvido nesta dissertação.

2.3 Hyper-G

Hyper-G [26] é um sistema de informação hipermídia desenvolvido na Graz University of Technology, Áustria. Projetado para ser um sistema de larga escala, multiusuário e distribuído, Hyper-G apresenta suporte à navegação hierárquica, a viagens guiadas e a facilidades de pesquisa, como a indexação por atributos dos documentos e a variação do escopo de busca. Além disso, provê meios para a manutenção e estruturação de documentos, edição interativa de elos e controle de acesso hierárquico.

A noção de hierarquia é obtida, acrescentando ao modelo hipertexto usual de nós e elos, coleções e grupos (*clusters*). Os documentos (a unidade básica de informação no Hyper-G) são agrupados em grupos, que por sua vez formam as coleções. Os grupos são usados para definir documentos agregados multimídia (um texto e um vídeo exibidos simultaneamente) e multilíngüe (versões em diferentes idiomas). As coleções podem fazer parte de uma ou mais coleções, recursivamente, criando a estrutura hierárquica.

Além da navegação por elos e pela estrutura, pode-se consultar documentos por meio de mecanismos de busca. Assim que são criados, todos os documentos são indexados automaticamente na base de dados do sistema, permitindo a procura pelo conteúdo ou por alguns atributos (título, autor, palavras-chave, data de criação, etc.). A busca pode ser limitada a uma coleção ou a um conjunto delas, em um único servidor Hyper-G ou em vários.

Os documentos criados no Hyper-G podem ser visualizados através de navegadores nativos: o Harmony para a plataforma X Windows Unix e o Amadeus para Microsoft

Windows. O navegador Harmony, visto na Figura 2.2, apresenta diversas ferramentas para orientar o usuário durante a navegação e auxiliar a localização de informações [3]:

- Um mapa gráfico local, que mostra todos os elos que partem e que chegam ao documento corrente;
- Uma lista com a hierarquia de coleções (*session manager*): uma interface ativa parecida com a dos gerenciadores de arquivos, a qual exibe uma estrutura em árvore das coleções, permitindo o acesso a documentos que não apresentam elos;
- Um histórico, que fornece uma lista dos mais recentes documentos visitados pelo usuário;
- Uma visão gráfica em 3D da estrutura de coleções (*information landscape*), na qual os usuários podem "voar" pelo espaço hipermídia para consultar os documentos.

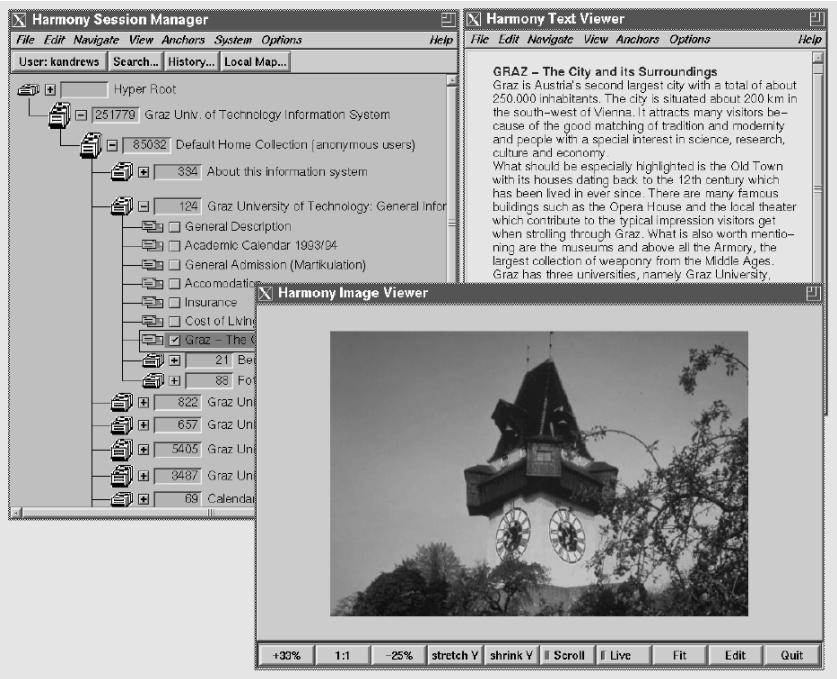


Figura 2.2 – O cliente Harmony do Hyper-G.

O mapa local, visto na Figura 2.3, apresenta uma visão de curto alcance (o padrão é mostrar dois níveis) das relações de referência do documento selecionado. O usuário pode navegar por este mapa até o próximo documento e ativá-lo com um clique duplo, atualizando o *layout* do mapa. As visões do mapa local, do gráfico em 3D e da lista de coleções são sincronizadas: a mudança em uma delas é automaticamente refletida nas outras.

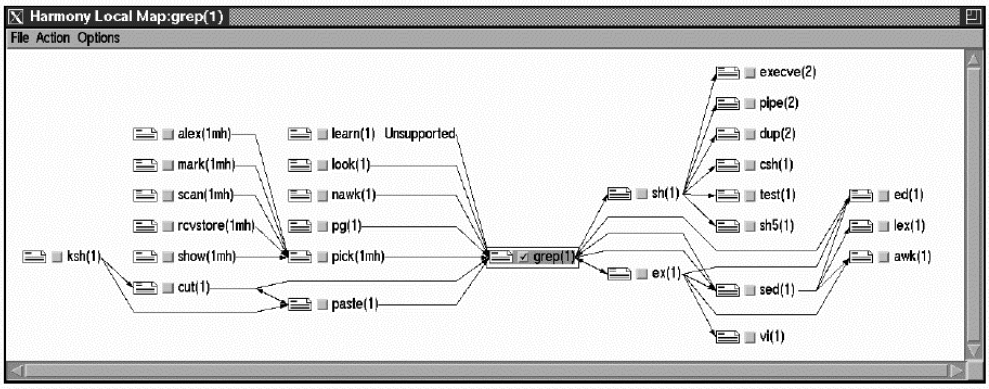


Figura 2.3 – O mapa local de elos do Harmony.

Na visão em 3D, as coleções são mapeadas em planos no espaço hipermídia e os documentos componentes de uma coleção, em blocos posicionados sobre o plano correspondente. A Figura 2.4 ilustra esta visão do espaço hipermídia. A cor e altura dos blocos codificam respectivamente o tipo e o tamanho dos documentos. As relações de inclusão entre coleções são visualizadas por meio de arestas entre os planos. Com o *mouse*, o usuário controla a direção e a velocidade do movimento no espaço. Na figura, vê-se em uma outra janela menor, uma visão geral da região do espaço exibida na janela principal, para manter a orientação do usuário.

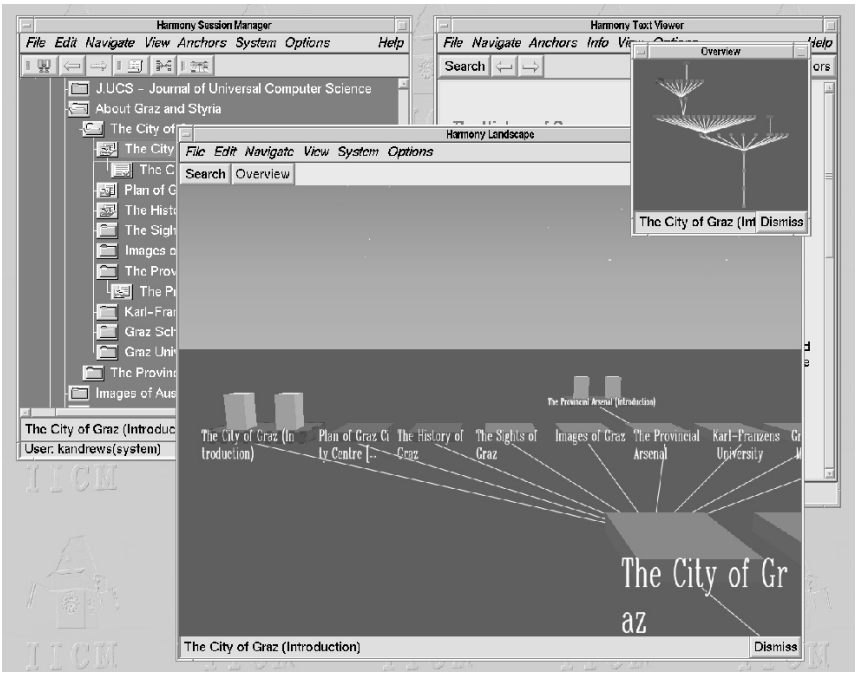


Figura 2.4 – O information landscape do Harmony.

Os clientes Hyper-G, além de serem usados para a navegação e busca de informações, apresentam ferramentas para a edição de documentos. Os usuários podem, desde que tenham permissão para tal, incluir documentos em coleções e criar elos entre documentos, interativamente. Como em todos os OHS, os elos não são armazenados nos documentos, mas em bases de dados separadas. Assim, não se restringem aos documentos texto, podem ser percorridos no sentido inverso e são atualizados automaticamente quando seus destinos são movidos ou apagados.

2.4 Kaomi

Kaomi [24] é uma ferramenta utilizada para projetar ambientes de autoria de documentos multimídia. A autoria é baseada em um conjunto de visões do documento, que são sincronizadas entre si pela seleção de objetos ou por instantes de tempo predefinidos.

Cada visão admite algum tipo de edição, que é automaticamente propagada para as outras visões a fim de garantir a consistência dos dados. A ferramenta é flexível o bastante para permitir extensões e/ou modificações em suas visões e suportar uma variedade de formatos declarativos de documentos, além de operar no ambiente heterogêneo da Internet.

As diferentes visões oferecidas pelo Kaomi proporcionam uma efetiva interface ao usuário para navegação e edição de documentos multimídia baseada no paradigma WYSIWYG[1]. A visão de apresentação, que pode ser considerada a principal, mostra a apresentação de um documento com as funções de controle básicas (tocar, parar, pausa, avançar, etc.). A visão de objetos revela o conjunto de objetos que compõem a organização lógica de um documento. Na estrutura hierárquica formada, as folhas representam os objetos básicos e os nós não terminais são os objetos compostos (uma cena, por exemplo). Diferentes filtros podem ser aplicados para organizar a exibição desse conjunto (ordem alfabética ou o tipo de mídia do objeto, por exemplo).

Uma outra visão exibe o arquivo fonte do documento em seu formato textual próprio. Finalmente, a visão temporal projeta em um gráfico a execução de um documento no tempo, mantendo a noção de sua estruturação hierárquica. Nesta visão, um documento é representado por um grafo: os arcos são associados aos objetos componentes do documento e os nós representam os pontos de início e fim destes objetos. A Figura 2.5 mostra a tela de um ambiente de autoria implementado com o Kaomi. Cada janela apresenta uma visão do documento. O autor pode executar a apresentação de um documento e, a qualquer momento, interrompê-la para editar um determinado elemento.

---

[1] Acrônimo para a expressão em inglês “What You See Is What You Get”, que caracteriza programas que geram documentos tal como são visualizados na tela do computador.

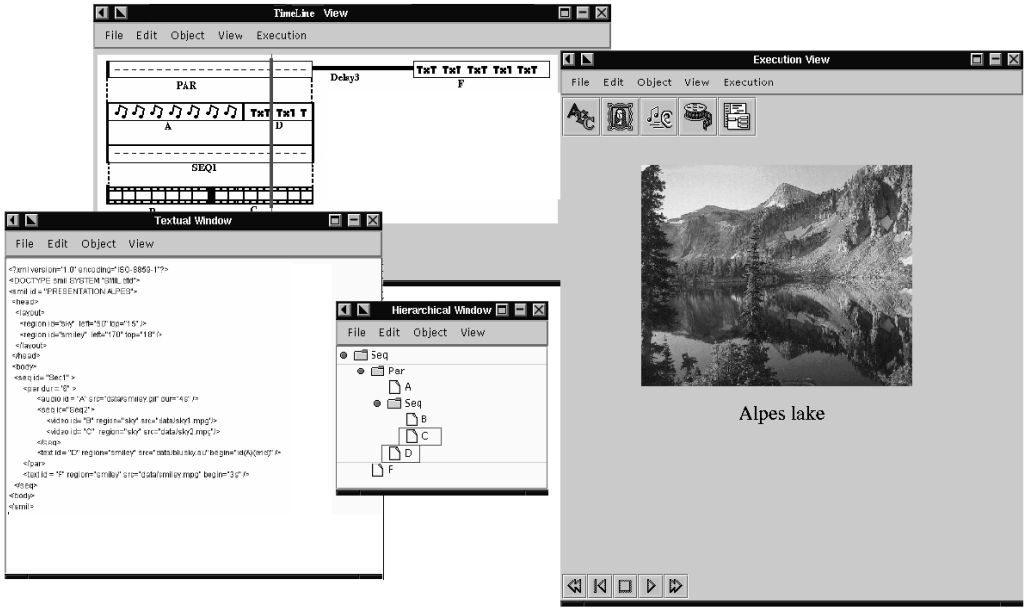


Figura 2.5 – Um editor Smil desenvolvido com o Kaomi.

A aplicação provê as funções usuais para a autoria de documentos (*abrir*, *fechar*, *criar* e *salvar*), além das operações de *editar propriedades*, *inserir*, *remover*, *copiar* e *colar* objetos em documentos. Relações de hipertexto (*goto*), espaciais (*alinhar*, *centralizar*, ...) e temporais (*exibir em série ou em paralelo*, *inserir atraso*, ...) também podem ser definidas. Quando um objeto é removido, suas relações espaciais e temporais com outros objetos também são apagadas. Se um objeto composto (nó pertencente à hierarquia do documento) for copiado/colado, todos os seus nós componentes e as relações entre eles também são afetados.

O ambiente de autoria produzido pelo Kaomi pode ser ajustado para se adequar ao contexto da aplicação. A ferramenta oferece meios de ser expandida, permitindo que novas visões sejam adicionadas, mantendo as características de edição e sincronismo. Por exemplo, pode ser criada uma visão que mostre a estrutura hipertexto dos documentos. É possível acrescentar ainda novas funções em uma visão, assim como definir novas relações espaciais e temporais e ampliar o suporte a novos tipos de formato de dados.

Para usar o Kaomi, o projetista deve desenvolver os métodos que convertem os documentos de sua linguagem fonte para a estrutura de dados utilizada pela ferramenta, e vice-versa. Assim, deve ser fornecido um analisador completo (sintático e semântico) dessa linguagem para que o Kaomi crie a estrutura lógica do documento e o grafo de execução da apresentação. Como padrão, a ferramenta já fornece suporte à linguagem XML. Em [24], os autores descrevem dois sistemas implementados com o Kaomi: um editor Smil e uma versão do ambiente de autoria Madeus.

2.5 Microsoft Front Page

Front Page [29] é a ferramenta da Microsoft para a criação e gerenciamento de *web sites*. Possui como principal característica a facilidade de uso, que permite a qualquer usuário pouco familiarizado com os padrões da *web* construir documentos WWW. Front Page é composto por um navegador, próprio para o projeto e administração de *sites*, e um editor, para a autoria de páginas HTML.

O navegador do Front Page é utilizado para a criação da estrutura e do *layout* dos documentos WWW. Com esta ferramenta, pode-se organizar os documentos em arquivos e diretórios, testar a consistência de seus hiperelos, administrar os privilégios de acesso, aplicar temas (conjuntos de padrões gráficos) em suas páginas e acionar o editor HTML. Quando um documento é finalizado, utiliza-se o próprio navegador para transferí-lo para um servidor WWW remoto.

A autoria do conteúdo dos documentos é realizada com o auxílio do editor do Front Page, podendo-se adicionar texto, formulários, imagens ou qualquer outro elemento, além de converter arquivos de vários formatos para o HTML. Embora seja uma ferramenta poderosa,

é bastante fácil de ser usada, devido principalmente a sua interface, familiar aos processadores de texto comuns. O editor exibe o conteúdo de uma página da mesma forma que é apresentado nos navegadores WWW. À medida que o documento é criado, a ferramenta insere automaticamente o código HTML necessário. Assim, a criação de elos, por exemplo, torna-se um processo simples, bastando apenas selecionar uma âncora no documento origem e o nome do documento destino. A Figura 2.6 mostra a tela do editor do Front Page exibindo o conteúdo de uma página HTML.

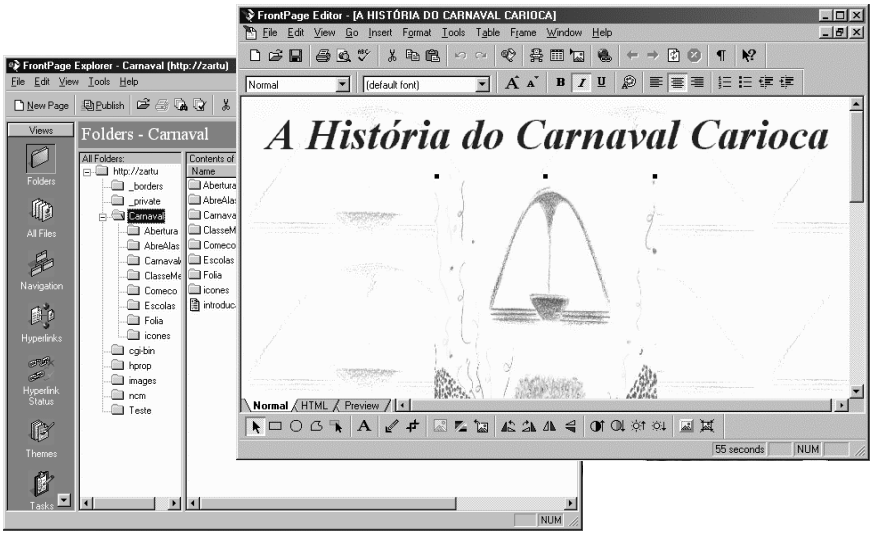


Figura 2.6 – O Editor HTML do FrontPage.

O Front Page apresenta, de uma forma integrada, diferentes visões dos documentos WWW. As duas visões principais são uma lista com todos os arquivos e uma estrutura hierárquica, em forma de árvore, dos diretórios onde estão organizados esses arquivos. A seleção de um nó nesta árvore causa a exibição dos nós contidos no diretório correspondente. Em todas as visões, pode-se criar, editar ou remover as páginas HTML apresentadas. A ferramenta apresenta suporte à autoria cooperativa: as visões estruturais dos documentos podem ser atualizadas para incorporar modificações realizadas por outros usuários.

Uma outra visão mostra a estrutura navegacional de um documento. Através do editor HTML, pode-se definir automaticamente relacionamentos entre páginas, compondo uma organização hierárquica de navegação do documento. Esta estrutura é exibida nesta visão. No desenho formado, pode-se selecionar um determinado nó a fim de expandir ou contrair sua sub-árvore, sendo possível também editar a própria estrutura navegacional. A Figura 2.7 ilustra um exemplo de um documento hipertexto composto por vários nós interligados hieraquicamente.

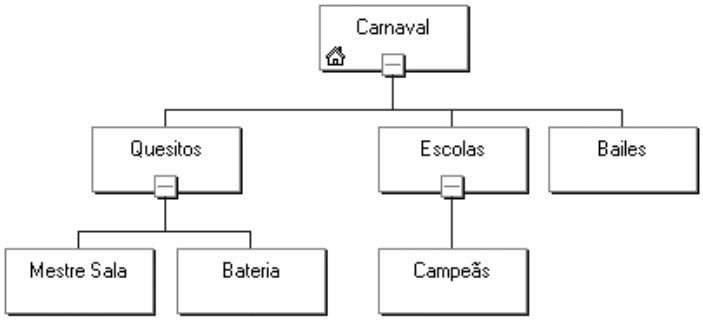


Figura 2.7 – Mapa Navegacional do Front Page.

A ferramenta possui também um mapa que apresenta as possibilidades de navegação de um determinado nó, mostrando os hiperelos que ancoram nele e seus nós adjacentes (Figura 2.8). A seleção de um documento de uma lista, localizada ao lado do mapa, provoca a atualização total do desenho. Este é inicialmente formado por um grafo com três camadas, que vai crescendo à medida que o usuário seleciona outros nós no mapa. Dependendo do nó escolhido, determinados ramos do grafo são fechados para garantir a legibilidade do desenho. Dentre as desvantagens observadas, nota-se que, como no Front Page não há distinção entre tipos de elos, os elos que formam a estrutura navegacional do documento também aparecem neste desenho. Além disso, como o grafo criado deve sempre formar uma árvore, se mais de um nó possuir elos para um mesmo nó, este será apresentado no mapa mais de uma vez.

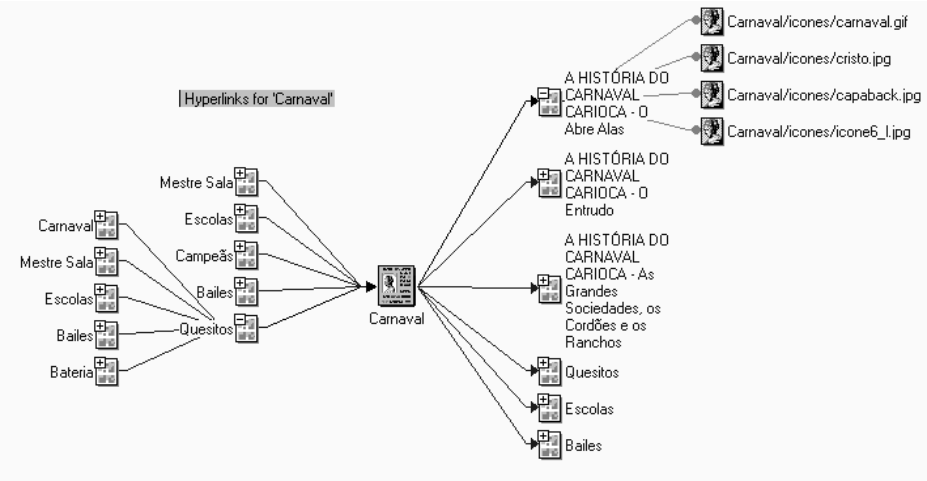


Figura 2.8 – Mapa de hiperelos do Front Page.

Capítulo 3

Sistema HyperProp

O sistema HyperProp é um sistema hipermídia aberto desenvolvido no Laboratório TeleMídia do Departamento de Informática da PUC-Rio. O sistema é baseado em um modelo conceitual de dados chamado Modelo de Contextos Aninhados – NCM (*Nested Context Model*) [48]. Neste capítulo, são introduzidas algumas das definições básicas deste modelo, importantes para o entendimento do sistema e do desenvolvimento deste trabalho. Em seguida, são analisados alguns aspectos da arquitetura e da interface gráfica do HyperProp. Uma descrição completa do NCM pode ser encontrada em [47]

3.1 Modelo de Contextos Aninhados

A definição de documentos hipermídia no NCM é baseada nos conceitos usuais de nós e elos. Os nós representam fragmentos de informação e os elos, relacionamentos de referência entre os nós que interligam. Basicamente, o modelo determina duas classes de nós: nós terminais e de composição. A Figura 3.1, extraída de [47], mostra o diagrama da

hierarquia de classes do NCM. Apenas as classes relevantes para esta dissertação serão apresentadas nesta seção.

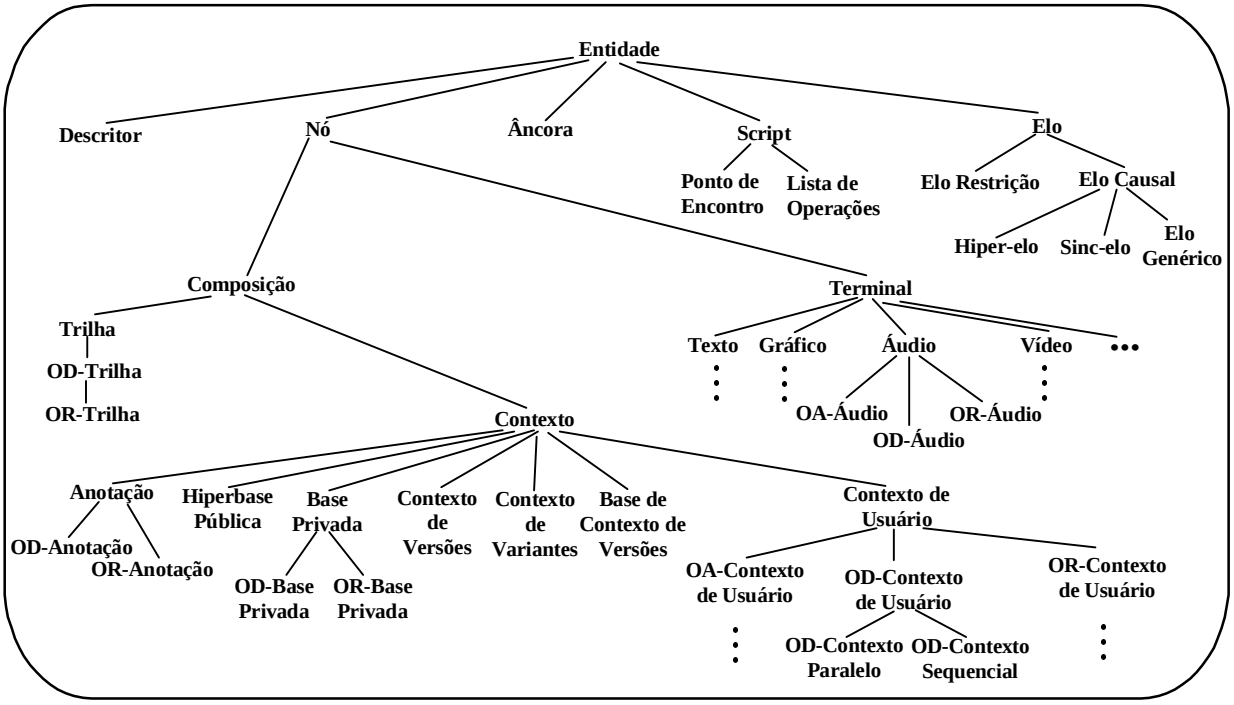


Figura 3.1 – Hierarquia de classes do NCM.

Uma *entidade* possui atributos como um identificador único, a data de criação e o autor. Um *nó* é uma entidade que tem como atributos adicionais um conteúdo e listas de âncoras e de descritores. O *conteúdo* de um nó é um conjunto de unidades de informação, que fazem parte da definição do nó e dependem de sua classe. A *âncora* define uma região marcada do conteúdo, oferecendo o acesso a segmentos do conteúdo do nó para a definição dos elos. O conjunto de âncoras age como uma interface externa do nó, impedindo que modificações no conteúdo de um nó se reflitam em outras entidades que o referenciam. O *descritor* contém as informações necessárias para a apresentação do nó no tempo e espaço, possibilitando que um mesmo objeto seja exibido de várias maneiras diferentes. Apenas um descritor da lista pode ser selecionado para cada instância do nó.

Um *nó terminal* é um nó cujo conteúdo é dependente da aplicação, sendo especializado em nós associados às diversas mídias (texto, imagem, vídeo, etc.). A *composição* é utilizada na estruturação lógica dos documentos: seu conteúdo é uma lista de nós (terminais e de composição, recursivamente), podendo um nó estar contido mais de uma vez na lista, e em várias listas diferentes[2]. No documento hipermídia da Figura 3.2, os nós *a*, *b*, *c*, *d* e *e* são terminais e os nós *f*, *g*, *h* e *i* são composições. Para identificar uma dada instância de um nó, é definido o conceito de *perspectiva* como sendo toda a seqüência de composições aninhadas que o contém. Na Figura 3.2, a perspectiva do nó *c*, por exemplo, é P_c= *(i, h, f, c)*.

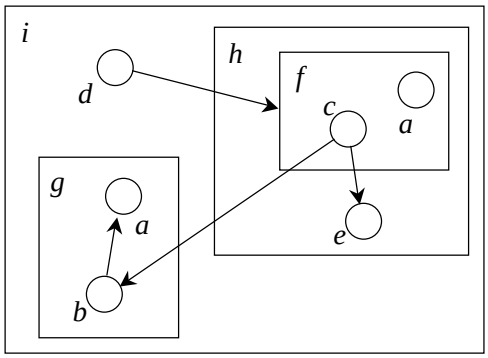


Figura 3.2 – Um hiperdocumento formado por composições aninhadas.

Um *nó de contexto* é um nó de composição que contém apenas nós terminais e nós de contexto, sem repetição de componentes. Pode ser especializado em outros tipos de nós, um dos quais é o nó contexto de usuário. Um nó *contexto de usuário* contém, além de nós, um conjunto de elos, ancorados no próprio contexto ou em nós recursivamente contidos nele. Com a definição dos elos nos nós de contexto, desvinculando-os dos nós terminais, obtém-se diferentes visões de um mesmo documento e permite o reuso dos dados sem a herança obrigatória das relações. Os nós contexto de usuário, quando especializados em nós de

[2] Apenas uma restrição é feita: um nó não pode estar recursivamente contido em si mesmo.

contexto seqüencial ou paralelo, também são úteis na definição da sincronização temporal de seus componentes.

Os *elos* caracterizam relações *n*:*m* entre nós, sendo definidos por conjuntos de pontos terminais de origem e destino, e por um ponto de encontro. Um *ponto terminal* é composto pela seqüência de composições que identifica cada nó relacionado pelo elo, pela âncora deste nó referida pelo elo e por um atributo que especifica o evento (de exibição, de seleção ou de atribuição)[3] associado à âncora. Os eventos definidos nos pontos terminais são relacionados pelo *ponto de encontro*, formado por uma condição e uma ação, de forma que a condição satisfeita no ponto de origem implica no disparo da ação a ela associada no ponto terminal destino.

No NCM, o tratamento de documentos multimídia é modelado a partir de três tipos de objetos nós: de armazenamento (OA), de dados (OD) e de representação (OR)[4]. Os nós OA são objetos permanentes e de acesso público, armazenados sempre em sua forma final, não podendo mais ser alterados. Um nó OD é criado como um novo nó ou como uma cópia local de um objeto de armazenamento, acrescida de atributos dependentes da aplicação e métodos para manipular tais atributos (exceto o conteúdo do nó). Um nó OR é uma cópia local de um objeto de dados, adicionando-se novos métodos para exibir e editar o conteúdo do nó (no caso de contexto, a estrutura de nós e elos). Estes métodos são derivados de um objeto descritor, sempre associado a um nó OR.

O descritor determina como um nó OR será exibido, por meio das especificações de iniciação e de término de sua apresentação ou edição, sendo dependentes da classe do nó a ser

---

[3] A definição de *evento* foge ao escopo desta dissertação, podendo ser encontrada em [47].

[4] Todos os objetos nós do NCM, independentes de sua subclasse, são objetos de armazenamento, ou de dados, ou de representação.

exibido. Quando um nó OR é apresentado, é definida uma ordem de preferência de descritores, pois pode haver vários descritores associados a ele. Um descritor especificado explicitamente pelo usuário sobrepõe-se ao descritor definido pelo elo para alcançar o nó. Esse, por sua vez, tem a preferência sobre o descritor selecionado no contexto que contém o nó, o qual é prioritário em relação ao atributo descritor do nó. Cada classe de nó possui um descritor padrão, que é usado se nenhuma das formas de identificação do descritor apresentadas for definida.

Especializações do nó de contexto, as classes contexto de versão, contexto de variantes, base de contextos de versões, hiperbase pública e base privada foram acrescentadas ao modelo com a finalidade de dar suporte ao versionamento de documentos e ao trabalho cooperativo. No NCM, podem ser criadas versões apenas de nós terminais e de nós de contexto de usuário, a partir de modificações de seus atributos versionáveis. Um atributo especificado como não versionável tem seu valor alterado sem a necessidade da criação de uma nova versão do nó.

O controle de versões deve prover meios para que o histórico de derivação dos nós seja registrado. Um *contexto de versões* é um nó cujo conteúdo agrupa as versões (objetos de dados ou de armazenamento) de uma mesma entidade, em algum nível de abstração, sem necessariamente implicar que uma versão foi derivada de outra. A *base de contexto de versões*, por sua vez, contém os contextos de versões de todos os nós. O *contexto de variantes* agrupa todos os objetos de representação derivados de um mesmo objeto de dados.

Entende-se por autoria cooperativa o processo de criar ou modificar uma hiperbase, ou um subconjunto da hiperbase, por um grupo de usuários. De forma geral, um ambiente de autoria cooperativa deve permitir que usuários compartilhem informação, oferecer alguma forma de criar informações privadas, e permitir a fragmentação do espaço global de trabalho

em frações menores para reduzir o espaço de navegação. No NCM, isto é obtido com os nós *hiperbase pública* e *base privada*.

Uma hiperbase é um conjunto de nós, tal que, se contém um nó de contexto, então todos os seus nós componentes também pertencem à hiperbase. Intuitivamente, a hiperbase pública do modelo agrupa todos os documentos persistentes (objetos de armazenamento), formando um repositório estável e compartilhado por todos os usuários do sistema, armazenado em um ou mais servidores NCM. As bases privadas se constituem num repositório de informações privadas de cada usuário, fragmentos do espaço global, contendo os documentos utilizados durante uma sessão de trabalho. Os nós da base privada são versões de nós trazidos da hiperbase pública, ou objetos criados durante a sessão de trabalho do usuário.

A hiperbase pública forma um único plano de objetos de armazenamento. As bases privadas, podendo ser inúmeras, formam vários planos de objetos de dados com seus respectivos planos de objetos de representação. A Figura 3.3, extraída de [47], mostra os três planos de objetos exemplificados respectivamente pela hiperbase pública, OD-base privada (especialização da base privada que contém apenas nós objetos de dados) e OR-base privada (que contém apenas nós OR).

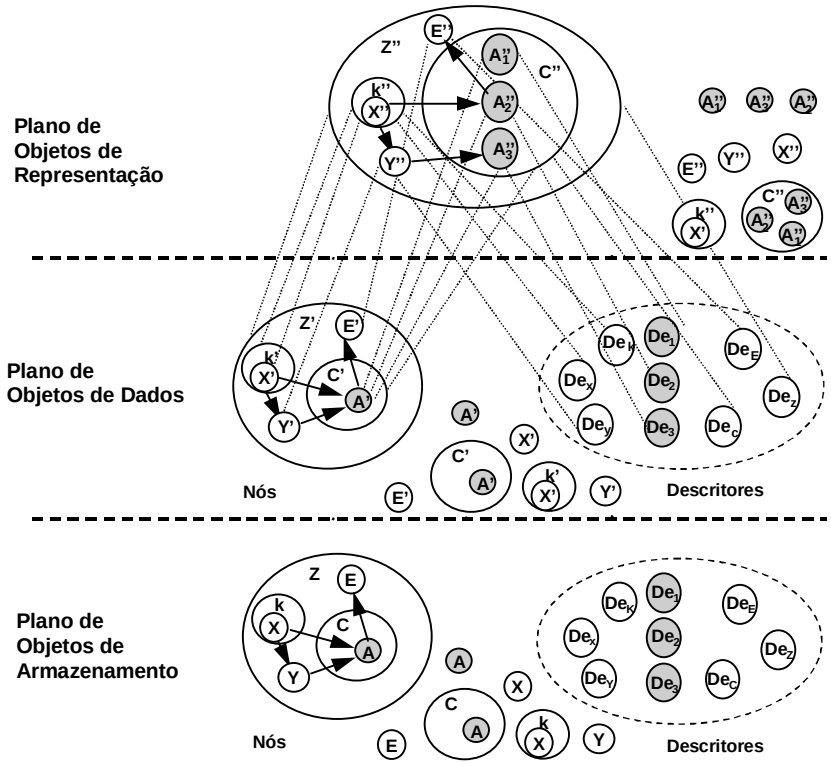


Figura 3.3 – Planos de Objetos de Armazenamento, Dados e Representação.

O suporte ao versionamento de documentos é dado através da manipulação dos nós entre os três planos de objetos. A criação de versões nas bases privadas é realizada pelas operações de *Check-out* e *Open*, que diferem entre si no tratamento de nós contexto de usuário. Quando o usuário quer mover um nó da base privada para a hiperbase pública, utiliza a primitiva *Check-in*. Se for um nó novo ou uma versão modificada, este é criado na hiperbase; caso contrário, ele é apenas eliminado da base.

A Figura 3.3 ilustra as relações de versionamento entre os nós, a qual um nó localizado num plano de objetos sempre descende de um nó do plano inferior. No plano de representação, diferentes versões de representação de um nó OD são formadas associando-se ao nó, descritores distintos. O nó OD *A'*, por exemplo, associado aos descritores De_1, De_2 e De_3, origina os nós OR $A"_1$, $A"_2$ e $A"_3$, respectivamente. Uma especificação detalhada do controle de versões no NCM pode ser encontrada em [47,49,36].

3.2 Arquitetura do Sistema HyperProp

O sistema HyperProp foi projetado para auxiliar a autoria e a distribuição de hiperdocumentos baseando-se nas facilidades oferecidas pelo modelo NCM. Em sua versão atual, implementada em Java, o HyperProp é uma aplicação cliente-servidor, cuja arquitetura é apresentada na Figura 3.4.

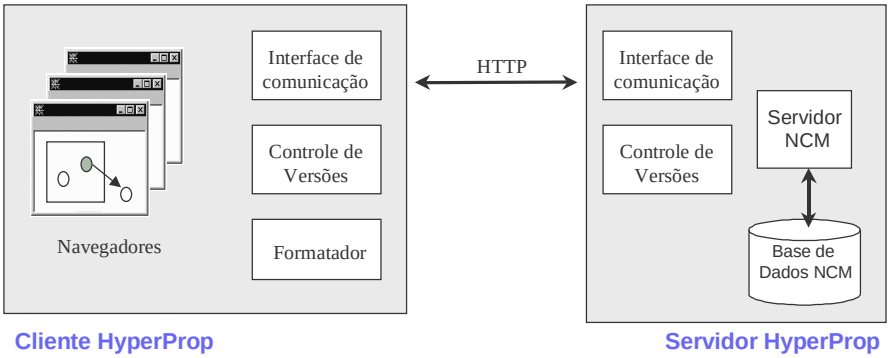


Figura 3.4 – Arquitetura Cliente-Servidor do Sistema HyperProp

O servidor, responsável pela persistência e gerência dos objetos NCM, possui uma conexão a um banco de dados orientado a objetos, onde são armazenados todos os dados relativos à estrutura hipertexto dos documentos contidos na hiperbase pública. O conteúdo dos nós não é mantido pelo sistema, podendo estar armazenado, de forma distribuída, em qualquer servidor da rede. O sistema busca o conteúdo a partir da URL associada a cada nó terminal. Maiores detalhes do servidor do sistema HyperProp podem ser obtidos em [31].

O módulo cliente do HyperProp, um programa também implementado em Java[5], é composto por navegadores que apresentam uma interface gráfica para a visualização da estrutura de documentos composta por uma visão, que apresenta as relações de hierarquia e referência dos nós do documento na forma de um grafo. O módulo cliente também oferece aos usuários, os ambientes de autoria e de apresentação (ou execução) de documentos. O ambiente de autoria possui ferramentas que permitem a criação de documentos NCM, através da edição de seus componentes, da definição de seus relacionamentos e da especificação de seu comportamento quando exibidos.

A exibição do conteúdo de um nó é determinada pelo formatador NCM, um módulo do sistema responsável pelo controle da navegação nos documentos. O formatador é composto por um compilador, um executor e um conjunto de controladores. A função do compilador é receber a especificação completa da apresentação de um documento, definida no ambiente de autoria, e gerar a estrutura de dados utilizada pelo executor para construir o plano de execução. O executor, com base nesses dados, instancia um controlador passando como parâmetros o nó, suas âncoras visíveis e seu descritor correspondente. Cabe ao controlador buscar o conteúdo do nó e exibí-lo, enquanto que o executor deve tratar os eventos gerados durante a apresentação do documento [44].

[5] A linguagem Java é a opção ideal para aplicações baseadas na Internet, pois seu código é executado independentemente da plataforma e sistema operacional utilizados.

3.3 Interface Gráfica do Sistema HyperProp

O sistema HyperProp oferece vários navegadores gráficos para auxiliar o usuário na localização, autoria e apresentação de documentos hipermídia. De acordo com o mapa hipertexto que apresentam, os navegadores podem ser de dois tipos: compostos e simples. Os primeiros possuem uma interface formada por uma visão que apresenta as relações de hierarquia e referência dos nós de um documento, sob a forma de um grafo composto. Além disso, possuem uma visão que exibe a estrutura em árvore das composições que fazem parte dos documentos. Outros navegadores mostram apenas um nível de aninhamento do conteúdo de uma composição, apresentando, portanto, apenas mapas formados por grafos simples.

São navegadores de grafos compostos, os navegadores que exibem a estrutura da hiperbase pública, da base de contextos de versões e da base privada. Os navegadores de contexto de versões e de contextos de variantes, utilizados no controle de versões, e o navegador de contexto, que mostra o conteúdo de uma composição no ambiente de apresentação, são exemplos de navegadores de grafos simples.

No grafo exibidos pelos navegadores, os elos aparecem como setas, conectando seus nós. Nós terminais são representados por ícones, de acordo com o seu tipo (texto, imagem,...). Ícones também figuram composições, que aparecem como pastas, podendo ser abertas ou fechadas. Ao ser aberta, uma composição passa a exibir seus componentes como um subgrafo. Esta idéia traduz bem os conceitos de modularidade e encapsulamento embutidos nas composições e evita uma sobrecarga de informações ao visualizarmos a estrutura de um documento. O *layout* do grafo é criado automaticamente, mas todos os nós podem ser rearranjados pelo usuário, de acordo com sua preferência.

A Figura 3.5 mostra uma sessão típica do sistema HyperProp, composta por navegadores que oferecem visões da estrutura dos documentos em diferentes níveis no sistema. Os navegadores de hiperbase pública e de base privada exibem respectivamente a estrutura do repositório público de documentos do servidor HyperProp e dos documentos manipulados localmente pelo usuário. Os nós trazidos para a base privada têm seu conteúdo apresentado por meio de navegadores próprios para cada tipo de mídia: no exemplo, vê-se um nó de composição e um nó texto sendo exibidos. Unir numa única interface gráfica a autoria e a navegação de documentos é uma característica importante do HyperProp. O ambiente de autoria do sistema aproveita o mapa estrutural para dispor uma forma simples e intuitiva de edição dos componentes de um documento para o usuário.

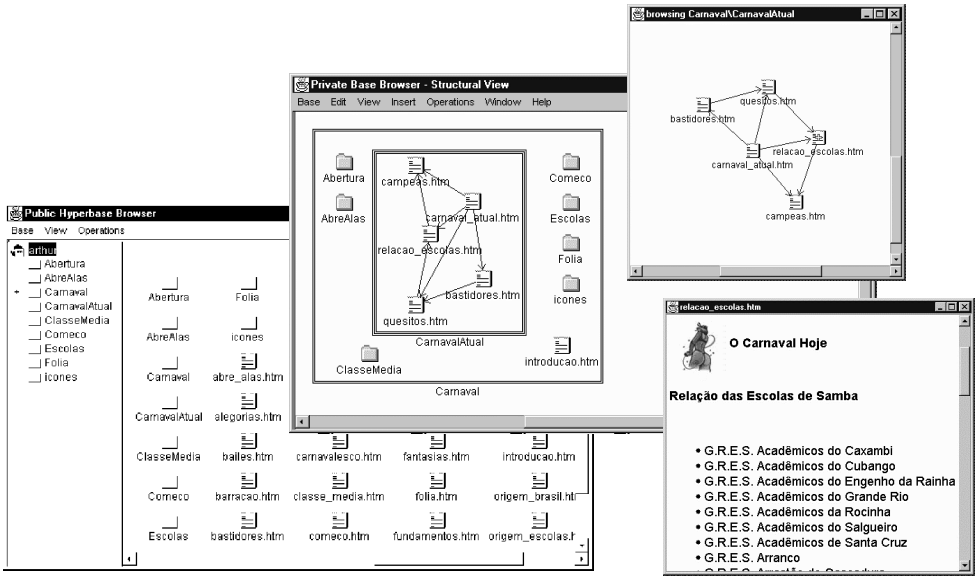


Figura 3.5 – Uma sessão típica do HyperProp.

No HyperProp, apesar das idéias básicas da interface gráfica de navegação e edição terem sido definidas nas versões anteriores do sistema, existiam importantes desafios na implementação de seus navegadores. Nos capítulos seguintes, estas questões são abordadas com mais profundidade. Em cada caso, o problema e a solução encontrada são revistos de maneira mais completa, descrevendo também detalhes da implementação realizada.

Capítulo 4

Navegação em Estruturas de Documentos Hipermídia

Um *grafo* é um conjunto finito *V* e um conjunto *E* de pares de elementos distintos de *V*. Os elementos de *V* são denominados vértices (ou nós) do grafo e os elementos de *E*, de arestas (ou elos) [51]. Cada aresta é expressa pelo par de vértices que a forma, sendo denominados nós adjacentes. Em muitos casos, definem-se grafos com alguma informação associada, para a modelagem de certas estruturas relacionais. Este trabalho está particularmente interessado na utilização de grafos para construir mapas de documentos hipertexto, onde os nós de um documento representam os vértices do grafo e os elos, as arestas.

Os navegadores são componentes fundamentais dos sistemas hipermídia, ao permitirem o acesso não-linear à informação e fornecerem uma visão geral do espaço hipermídia. Deve ser função do navegador mostrar um mapa de todo o documento ou de parte dele, como forma de reduzir a desorientação e a sobrecarga cognitiva dos leitores, oferecendo uma importante medida de contexto e espaço. Apresentar o espaço de informações como um

grafo não é simples, pois este precisa ser desenhado de maneira clara para que auxilie a navegação do usuário. A utilização de algoritmos para a disposição automática do desenho (*layout*) de grafos devem evitar, por exemplo, que dois nós se sobreponham, que haja vários cruzamentos de elos, etc.

Neste capítulo, é apresentada a interface gráfica da ferramenta de navegação do HyperProp, indicando as modificações que foram realizadas na biblioteca de desenho de grafos empregada. O algoritmo de *layout* utilizado pela ferramenta é analisado em detalhes, assim como as soluções encontradas para adaptá-lo ao desenho de documentos hipertexto. Inicia-se o estudo com uma introdução ao problema de desenho de grafos.

4.1 Desenho de Grafos

O desenho de grafos pode ser definido como a transformação de um grafo em uma representação visual, chamada de diagrama ou desenho (veja Figura 4.1). Os diagramas associados a um grafo são úteis desde que efetivamente transmitam alguma informação para quem for analisá-los. Por isso, é importante construí-los de maneira adequada, pois um diagrama ruim pode confundir mais o leitor do que orientá-lo.

O problema principal em se desenhar automaticamente um grafo é projetar um algoritmo que estabeleça uma posição para cada vértice e uma rota para cada aresta. Este problema tem sido alvo de intensa pesquisa desde o advento das estações gráficas de trabalho, no início da década de 80. Uma bibliografia [6], publicada em 1994, lista mais de 300 estudos sobre o problema, que também é tema de conferências anuais.

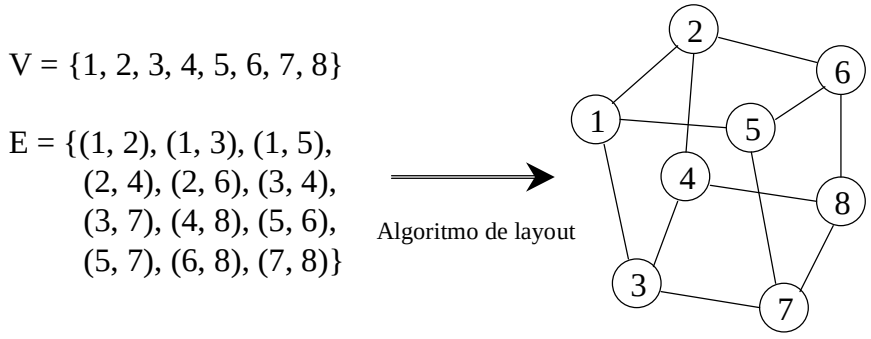


Figura 4.1 – O desenho de grafos.

Di Battista *et al.* classificou os requisitos do desenho de um grafo em três categorias: *convenções*, *restrições* e *estéticas* [7]. As convenções são as regras básicas para o desenho, como o espaço a ser utilizado (normalmente um retângulo no plano R^2), ou o grafo a ser desenhado: se este deve ser planar, direcionado ou não, ortogonal, com elos em linha reta ou apresentando quinas, etc. Normalmente, existem algoritmos de *layout* específicos para cada classe de grafos. As restrições geralmente referem-se à disposição absoluta ou relativa dos vértices de um grafo. Por exemplo, um grafo pode possuir uma semântica que determina que um dado subconjunto de vértices deva ser desenhado em um retângulo que não possa incluir nenhum outro vértice.

É possível, usando apenas convenções e restrições, determinar completamente um desenho. Pode-se, por exemplo, projetar um algoritmo simples para visualização de um grafo não-direcionado que disponha aleatoriamente os vértices num espaço finito e desenhe as arestas como linhas retas ou apresentando quinas para evitar a sobreposição com outros elementos no desenho. O resultado, entretanto, não levaria em consideração a legibilidade do grafo resultante, nem os dados que ele representa. Conseqüentemente, os desenhos gerados por esta estratégia poderiam não ser muito informativos.

Os critérios estéticos especificam os objetivos a serem otimizados para tornar um desenho legível. A escolha de uma determinada qualidade estética é altamente subjetiva e

dependente do propósito do desenho. Os principais critérios para um desenho de um grafo são [6]: minimizar o número de cruzamento de elos, distribuir uniformemente os vértices, desenhar a maioria das arestas dirigidas num mesmo sentido, minimizar o número de quinas nas arestas, realçar as simetrias do grafo, desenhar agrupamentos de vértices como regiões convexas, minimizar a área do desenho, maximizar a resolução angular entre arestas num mesmo vértice, etc.

Gerar desenhos esteticamente agradáveis é um problema complicado, pois em muitos casos não é possível satisfazer dois critérios simultaneamente. Por exemplo, a Figura 4.2 mostra dois desenhos de um mesmo grafo. O primeiro desenho minimiza o cruzamento de arestas, enquanto que a simetria é otimizada pelo segundo. Embora algumas vezes excludentes, critérios estéticos podem ser combinados, por meio de um grau de prioridade atribuído a cada um deles, para formar algoritmos mais complexos. Através da quantificação e ajuste destes critérios, chega-se à formulação de problemas de otimização numérica para a composição de desenhos de grafos, que só são possíveis de serem resolvidos com base em heurísticas.

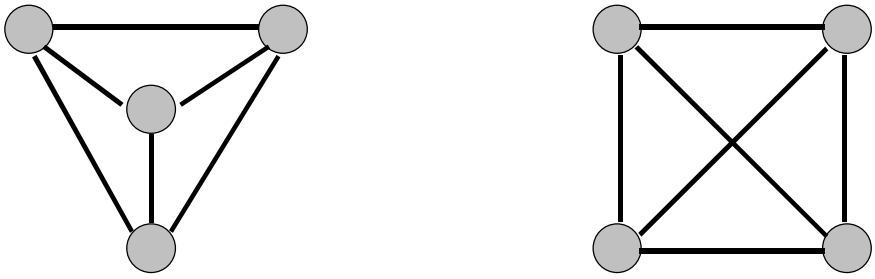


Figura 4.2 – Dois desenhos de um mesmo grafo com critérios estéticos diferentes.

À medida que a quantidade de informação que se quer exibir torna-se maior e as relações, mais complexas, o modelo clássico de grafo tende a ser inadequado. Em muitas aplicações, como por exemplo os navegadores de documentos hipermídia, onde a informação estrutural das entidades também precisa ser visualizada, um simples diagrama formado por nós

e elos não é suficiente para representar tais estruturas, sendo portanto necessária a criação de modelos mais poderosos de grafos.

Um dos modelos não-clássicos de grafo é o *hipergrafo* [23]. Em um grafo comum, o elo expressa uma relação entre apenas duas entidades; já nos hipergrafos, um *hiperelo* pode significar relações associadas a um conjunto de entidades. Um outro modelo de grafo, que pode ser utilizado em várias aplicações, é o *grafo agrupado* (*clustered graph*) [17]. Este modelo consiste em um grafo não-direcionado *G* e um particionamento recursivo dos vértices de *G*, com cada parte formando um conjunto de vértices. No diagrama deste grafo, cada conjunto é desenhado como uma região fechada, podendo estar organizado em vários níveis. A Figura 4.3 mostra em (a) um exemplo de um hipergrafo e em (b) um grafo agrupado.

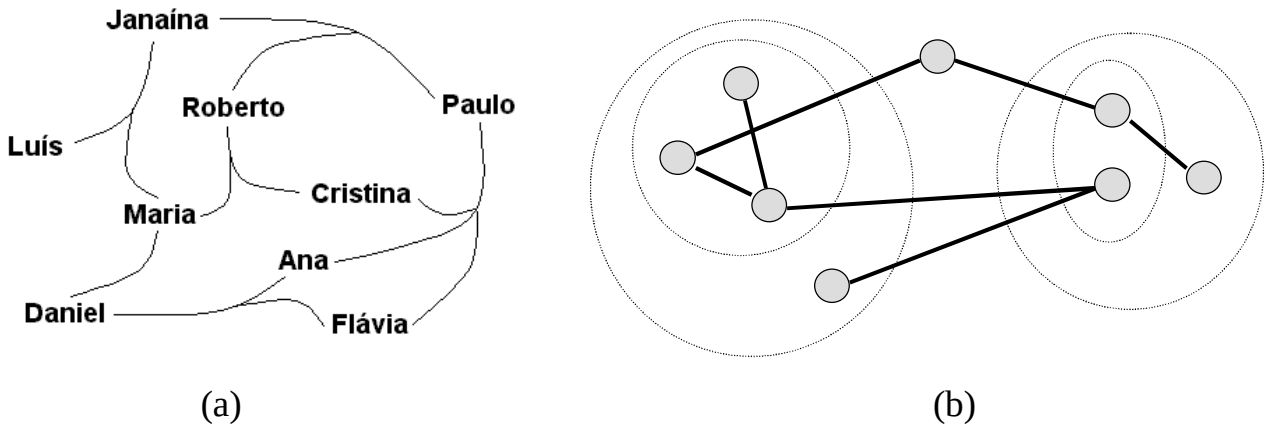
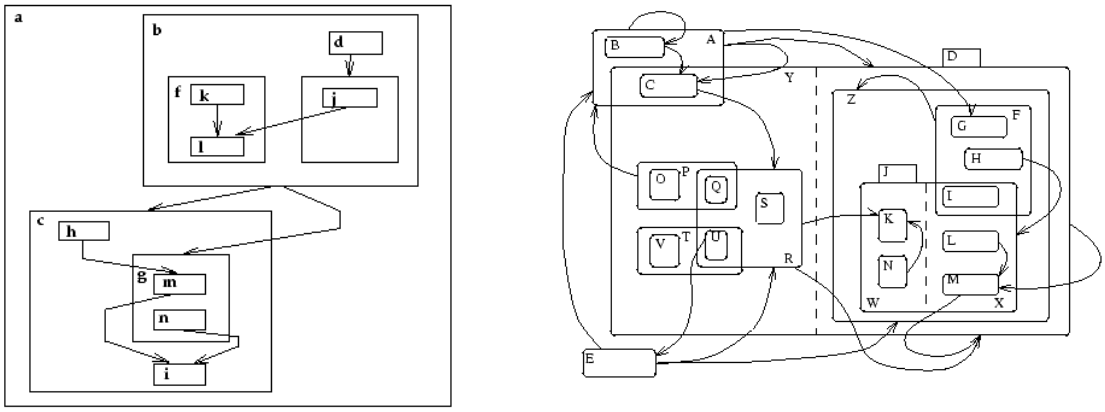


Figura 4.3 – Exemplos de modelos não-clássicos de grafos. (a) um hipergrafo. (b) um grafo agrupado.

Sugiyama e Misue [50] propõem um modelo chamado *dígrafo composto* (*compound digraph*), que permite relações de inclusão e adjacência entre nós. Embora o diagrama formado por este modelo também seja multinível, como no caso dos grafos agrupados, sua estrutura hierárquica é resultado do aninhamento de vértices especiais, chamados compostos, e não de sub-regiões do grafo. Um exemplo de um dígrafo composto é apresentado na Figura 4.4 (a).

Baseado nos hipergrafos e nos conceitos dos diagramas de Venn, Harel [21] apresenta um modelo mais genérico, chamado *grafo hierárquico* (*higraph*). Este tipo de grafo pode representar estruturas mais complexas, pois além de admitir hiperelos entre quaisquer nós, seus conjuntos de nós podem denotar relações de inclusão, interseção e ortogonalidade entre os elementos que contêm (Figura 4.4 (b)). Os grafos hierárquicos são apropriados para uma gama de aplicações, de modelos de bancos de dados a diagramas de estado de sistemas concorrentes [21].



(a) (b)

Figura 4.4 – Exemplos de modelos não-clássicos de grafos. (a) um dígrafo composto. (b) um grafo hierárquico.

Alguns algoritmos para o desenho automático de dígrafos compostos foram desenvolvidos [50,30]. Uma versão anterior dos navegadores de estrutura do sistema HyperProp, proposta em [31], era baseada no pacote D-ABDUCTOR da Fujitsu Laboratories, que implementava um destes algoritmos. O *software* oferecia uma interface gráfica para a construção de mapas na forma de grafos compostos, quase totalmente adequados ao modelo NCM. As restrições residiam na própria definição dos dígrafos compostos: além de permitir somente elos de referência simples (1:1), estes não poderiam conectar nós ancestrais e descendentes de relações de inclusão.

O algoritmo utilizado no pacote D-ABDUCTOR assumia que os elos de adjacência entre os nós eram direcionados, forçando uma ordenação nos vértices, para que os elos ficassem sempre no mesmo sentido. Embora as relações de referência dos sistemas hipertexto tenham a noção de origem e destino, em termos do desenho, este fator não é importante. Esteticamente, se considerarmos o grafo como sendo não-direcionado, o desenho resultante poderá ser mais legível e simétrico.

O algoritmo de *layout* dos mapas do espaço hipermídia, apresentados pelos navegadores do HyperProp, deve ser baseado em um modelo de grafo que retrate fielmente os objetos definidos no modelo NCM. Como o grafo que representa os nós hipertexto pode ser cíclico ou não, planar ou não, muito denso ou esparso, totalmente conexo ou não, etc.., o algoritmo de *layout* também deve ser suficientemente genérico para produzir bons desenhos para toda esta variedade de grafos.

Ao ser tratada como um grafo, a estrutura dos documentos baseados no modelo NCM é considerada complexa, devido principalmente à definição bastante ampla dos nós e elos do modelo, que permite, por exemplo, o aninhamento de nós de composições e o relacionamento multiponto-a-multiponto (*n*:*m*) entre quaisquer nós. Como, exceto o complexo grafo hierárquico, nenhum dos tipos de grafos apresentados se adapta completamente às especificações do NCM, um outro modelo de grafo teve de ser proposto[6]. Este novo modelo estende a definição dos grafos compostos.

O modelo de grafo composto estendido C é definido pela tripla $C = (V, E, F)$, formada pela composição dos grafos $D_1 = (V, E)$ e $D_2 = (V, F)$, que representam as relações de inclusão e adjacência em um conjunto finito de vértices V, expressas respectivamente pelos

[6] Na verdade, este modelo pode ser considerado um caso particular do grafo hierárquico, sem suas relações de ortogonalidade e interseção entre conjuntos de nós.

conjuntos finitos de arestas E e F. Os elementos de E são definidos pelo par (u, v), significando *u inclui v*, para $u, v \in V$. O grafo $D_1 = (V, E)$ é necessariamente uma árvore, chamada árvore de inclusão de C, cujos vértices são os nós compostos de C e folhas são os nós simples de C. Os nós que são contidos em um nó composto formam um subgrafo (que também pode conter nós compostos), desenhado dentro do nó que o contém. Os elementos de F são definidos pela n-tupla (u_0, u_1, ..., u_n, v_0, v_1, ..., v_m), significando que v_i é *adjacente* a u_j para u_j, $v_i \in V$, com $i \in [0, m]$ e $j \in [0, n]$, e $n \geq 0$, $m \geq 0$.

Como se vê, o modelo de grafo em que se baseia a interface gráfica da ferramenta de navegação do HyperProp aproveita a hierarquia de nós do modelo do dígrafo composto, os hiperelos do hipergrafo e a natureza não-dirigida dos grafos agrupados para modelar documentos hipertexto definidos pelo NCM, sem ser tão genérico quanto o grafo hierárquico.

Na implementação do sistema, foi utilizada uma biblioteca de desenho de grafos, desenvolvida em Java, para a criação dos mapas ativos. Como será visto mais adiante neste capítulo, esta biblioteca foi adaptada para suportar o modelo de grafos definido.

Uma vez definido o modelo de grafo adotado, resta determinar um algoritmo apropriado para o *layout* deste grafo. Como já mencionado, a estrutura de um documento hipertexto não tem restrições e, portanto, o grafo associado pode ter várias configurações. As únicas características asseguradas são que ele é não-direcionado e possivelmente composto (apresenta subgrafos). O algoritmo de *layout* deve levar em conta estas características e o caráter genérico do grafo para criar desenhos adequados a qualquer situação.

4.2 Modelo *Spring-Embedder*

Há várias estratégias diferentes que podem ser empregadas para desenhar um grafo genérico não-direcionado. Uma delas, por exemplo, é desenhar o seu equivalente planar, que pode ser obtido através de um simples teste de *planarização*. Se o grafo não for planar, técnicas podem ser utilizadas para torná-lo assim. A planarização de grafos é um problema NP-difícil, mas pode ser computada por meio de heurísticas em um tempo $O(n^2)$ ou menor [13]. Um outro método é atribuir um sentido às arestas do grafo, usando, por exemplo, o caminho euleriano, e aplicar um algoritmo para o desenho de grafos direcionados [46]. Finalmente, existe a família de heurísticas *force-directed*, que transformam os vértices e arestas de um grafo em um sistema de forças e encontram o estado de energia mínima deste sistema para determinar o melhor *layout* do grafo.

A abordagem dos algoritmos *force-directed* para o problema de desenho de grafos é expressar as preferências estéticas como forças que atuam sobre os vértices de um sistema físico análogo ao grafo, determinando o gradiente negativo da energia deste sistema ou de uma função implícita. A característica fundamental destes algoritmos é que a proximidade entre vértices no sistema deve corresponder à proximidade dos nós no desenho do grafo. O resultado final é obtido, resolvendo-se equações diferenciais ou executando-se simulações do sistema de forças.

Alguns dos critérios estéticos para o desenho de grafos não-direcionados são a simetria na distribuição dos vértices, a minimização de cruzamentos de arestas e o uso de arestas de comprimento uniforme [6]. A otimização destes critérios pelos algoritmos de *layout* é tipicamente um problema NP-completo ou NP-difícil, o que sugere a utilização de heuríticas. Os algoritmos da família *force-directed* variam principalmente na escolha do modelo físico que rege o sistema ou nas estratégias de otimização que calculam o estado final do sistema de

acordo com o modelo adotado. O *spring-embedder*, originalmente proposto por Eades em 1984 [16], é o mais popular algoritmo desta família.

No modelo *spring-embedder*, as arestas do grafo são substituídas por molas e os vértices, por anéis. Este sistema mecânico oscila até se estabilizar em um estado de energia mínima, quando a soma das forças em cada vértice é igual a zero. Em outras palavras, o desenho resultante será equivalente à configuração do sistema quando sua energia potencial total convergir para um valor localmente mínimo. As molas atuam nos dois vértices por elas conectados, com uma força[7] logarítmica. Em contrapartida, os vértices são carregados com cargas elétricas positivas, de modo a se repelirem com uma força proporcional ao inverso do quadrado de suas distâncias. A Figura 4.5 ilustra este modelo físico.

O algoritmo de *layout* automático estabelece randomicamente uma configuração inicial para o sistema e em seguida realiza um número fixo de iterações, na qual todos os vértices são movidos simultaneamente sob o efeito das forças exercidas sobre eles, até atingirem o estado de equilíbrio. Eades afirma que o algoritmo produz bons desenhos para vários exemplos de grafos, exceto para aqueles muito densos ou com um número pequeno de pontes. Também mostra que seu algoritmo possui um tempo de processamento aceitável para grafos com menos de cinqüenta vértices [16].

---

[7] Deve-se ressaltar que as "forças" neste algoritmo não causam aceleração. Não há energia cinética ou momento neste modelo físico; mais exatamente, a cada iteração a energia potencial do sistema é reduzida. Como conseqüência, o sistema pode ser descrito por meio de equações diferenciais de primeira ordem, ao invés de equações de segunda ordem.

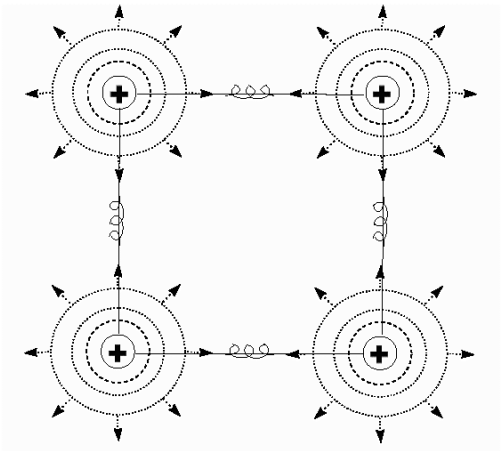


Figura 4.5 – O modelo spring-embedder.

A interface gráfica do sistema HyperProp utiliza como algoritmo de *layout* dos mapas de hiperdocumentos uma extensão ao algoritmo *spring-embedder*, desenvolvida por Kamada e Kawai [25]. A nova proposta modifica o modelo de Eades, eliminando as cargas elétricas dos vértices e associando uma mola a todos os pares de vértices do grafo, e não apenas para os pares que originalmente possuem arestas. Entre os vértices não adjacentes, contudo, o comprimento das molas é bem superior àquelas associadas aos vértices adjacentes.

As molas agem conforme a Lei de Hooke: a força exercida nos vértices é proporcional à diferença entre o comprimento em repouso da mola ($\mathcal{L}$) e a distância real entre os vértices. Se esta distância for maior do que $\mathcal{L}$, a mola atrai os dois vértices para mais perto; se a distância entre eles for menor, então a mola os impulsiona para mais longe. Portanto, vértices adjacentes são mantidos próximos uns aos outros por molas curtas, enquanto que molas mais longas mantêm vértices não adjacentes distantes entre si, ao mesmo tempo que limitam o tamanho total do sistema.

Para cada par de vértices, Kamada e Kawai fazem com que o comprimento em repouso da mola associada seja proporcional ao menor caminho entre os dois vértices no grafo, e que a rigidez da mola seja inversamente proporcional a este comprimento.

Conceitualmente, o processo de otimização deste modelo é tratado em termos da energia do sistema, em vez de se utilizar diretamente as forças que atuam sobre os vértices: a Lei de Hooke é integrada para obter-se a energia potencial para cada mola, que é quadrática em relação à diferença entre $\mathcal{L}$ e a distância real entre os dois vértices.

A energia total do sistema é representada pelo seguinte somatório [25]:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} \left(\left| p_i - p_j \right| - l_{ij} \right)^2 \text{ , onde}$$

p_1, p_2, ..., p_n são as posições das partículas no sistema que representam os vértices v_1, v_2, ..., v_n. l_{ij} é o comprimento natural da mola entre v_i e v_j, sendo definido como $l_{ij} = L \times d_{ij}$, onde L é o comprimento desejado (final) da aresta e d_{ij} é o comprimento do menor caminho entre v_i e v_j no grafo. Em geral, L é escolhido em função do diâmetro do grafo e da área disponível para o desenho. k_{ij} é a força da mola entre os vértices v_i e v_j, sendo definida como $k_{ij} = k / d_{ij}^2$, onde k é a constante elástica da mola.

Reescrevendo a equação da energia acima em termos das coordenadas xy, obtém-se:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} \left((x_i - x_j)^2 + (y_i - y_j)^2 + l_{ij}^2 - 2l\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \right)$$

A abordagem de Kamada e Kawai difere da proposta de Eades no cálculo do algoritmo de minimização. Ao invés de mover todos os vértices simultaneamente, este algoritmo desloca somente um vértice de cada vez no desenho. Para determinar o mínimo local do somatório acima, é preciso calcular suas derivadas parciais e encontrar o mínimo para cada vértice, resultando em $2n$ equações dependentes não-lineares. Como este sistema de equações seria difícil de ser resolvido, considera-se uma partícula por vez. A cada iteração, a partícula do sistema com maior energia é identificada e movida para um ponto estável com baixa energia. Seja esta partícula p_m, sua função de energia é dada por:

$$\Delta_m = \sqrt{\left(\frac{\partial E}{\partial x_m}\right)^2 + \left(\frac{\partial E}{\partial y_m}\right)^2}$$

p_m é movida gradativamente com o objetivo de minimizar Δ_m. O ponto final corresponde ao local onde a condição $\frac{\partial E}{\partial x_m} = \frac{\partial E}{\partial y_m} = 0$ é satisfeita. Este ponto é obtido resolvendo-se iterativamente, através do método de Newton-Raphson, as duas equações lineares abaixo, para δx e δy, e adicionando-as a x_m e y_m respectivamente na equação acima. Estes passos são repetidos até a função de energia parar de decrescer.

$$\frac{\partial^2 E}{\partial x_m^2}\left(x_m^t, y_m^t\right)\delta x + \frac{\partial^2 E}{\partial x_m \partial y_m}\left(x_m^t, y_m^t\right)\delta y = \frac{-\partial E}{\partial x_m}\left(x_m^t, y_m^t\right)$$

$$\frac{\partial^2 E}{\partial x_m \partial y_m}\left(x_m^t, y_m^t\right)\delta x + \frac{\partial^2 E}{\partial y_m^2}\left(x_m^t, y_m^t\right)\delta y = \frac{-\partial E}{\partial y_m}\left(x_m^t, y_m^t\right)$$

onde t é a iteração corrente. Os detalhes matemáticos do processo de otimização podem ser encontrados em [25,27].

O algoritmo *spring-embedder* pode ser resumido pelo pseudocódigo abaixo [46]:

1. computar d_{ij} para $1 \leq i \neq j \leq n$;
2. computar l_{ij} para $1 \leq i \neq j \leq n$;
3. computar k_{ij} para $1 \leq i \neq j \leq n$;
4. iniciar $p_1, p_2, ..., p_n$
5. while (max $\Delta_i > \varepsilon$)
6. faça p_m a partícula cuja energia satisfaça $\Delta_m = \max \Delta_i$;
7. while ($\Delta_m > \varepsilon$)
8. compute δx e δy ;
9. $x_m \leftarrow x_m + \delta x$
10. $y_m \leftarrow y_m + \delta y$
11. end while
12. end while

Além de ser considerado uma solução conceitualmente elegante, o algoritmo de Kamada e Kawai produz bons desenhos, pois reduz o número de cruzamento de arestas, impede a sobreposição de vértices e mostra simetrias presentes no grafo. Um mesmo grafo,

com diferentes configurações iniciais, converge para o mesmo desenho. Assim como grafos com estruturas similares também são desenhados de forma parecida.

A principal desvantagem desta abordagem é computacional: o modelo adotado requer um pré-processamento que encontra o menor caminho entre todos os pares do grafo. Esta fase consome um tempo $O(n^3)$ e um espaço $O(n^2)$, o que praticamente invalida seu uso em grafos maiores. Algoritmos mais eficientes para o cálculo do menor caminho devem ser usados para diminuir este limite, como por exemplo, o algoritmo de Dijkstra [11], cujo tempo de processamento é $O(nm\log n)$[8] para grafos esparsos. A biblioteca de desenho de grafos utilizada pelo HyperProp implementa o *spring-embedder* com o algoritmo de Dijkstra.

No algoritmo principal, $O(n)$ é o tempo necessário para *loop* interno computar Δ_m, δx e δy em cada iteração. O tempo requerido para o processo de minimização de energia é $O(Tn)$, onde T é o número total de *loops* internos. T é difícil de ser caracterizado, pois depende do valor da precisão da convergência (ε), da posição inicial dos vértices, do tamanho e da conectividade do grafo.

4.3 Implementação

Dentre os sistemas de desenho de grafos pesquisados, decidiu-se utilizar uma ferramenta desenvolvida pela Universidade de Auburn, EUA, para o desenho e *layout* de

[8] Onde n é o número de nós e m o número de elos do grafo.

grafos. Esta ferramenta, chamada VGJ (*Visualizing Graphs with Java*)[9], consiste em um conjunto de classes implementadas em java que constrói desenhos de grafos a partir de uma descrição textual ou criados pelo editor de grafos que compõe a ferramenta. A biblioteca VGJ possui uma interface gráfica completa para a edição e visualização do desenho, apresentando algoritmos de *layout* para diferentes categorias de grafos. Optou-se por usar a biblioteca VGJ porque ela implementa um algoritmo de *layout* adequado para o desenho de documentos hipertexto, como explicitado na Seção 4.2, e porque possui a idéia de que um grupo de nós possa ser representado por um único nó (lembrando a noção de composição do NCM), além de disponibilizar os fontes das classes e ser escrita em java – como a atual versão do sistema HyperProp.

A disponibilidade dos fontes é importante, uma vez que a biblioteca terá de ser adaptada para suportar totalmente o modelo NCM. Por exemplo, o NCM introduz o conceito de composições aninhadas na estruturação dos documentos hipermídia. No modelo de grafo utilizado pela biblioteca, um subgrafo pode ser encapsulado por um nó, mas este ao ser "aberto", some do diagrama. Ou seja, embora exista a idéia de estruturação hierárquica, a ferramenta não permite o desenho do grafo composto. Isto foi modificado para que o nó de composição, quando aberto, seja representado por um retângulo que contenha todo o subgrafo, incluindo até elos ancorados na própria composição.

Outras alterações nas classes do VGJ foram motivadas pela definição genérica dos elos no NCM. Como o modelo admite mais de uma relação multiponto-a-multiponto (*n*:*m*) entre nós, isto teve que ser implementado pela ferramenta. Os elos *n*:*m* são criados com o auxílio de nós *dummies* que servem como ponto de encontro do elo. Destes nós, partem e

[9] A documentação sobre o projeto, assim como o pacote com as classes da biblioteca podem ser encontrados na Internet, no endereço http://www.eng.auburn.edu/department/cse/research/graph_drawing/graph_drawing.html

chegam todos os ramos do elo A multiplicidade de elos entre dois nós é obtida através de um rótulo, que indica o número de elos sobrepostos, originalmente definidos assim ou resultantes do encapsulamento das composições.

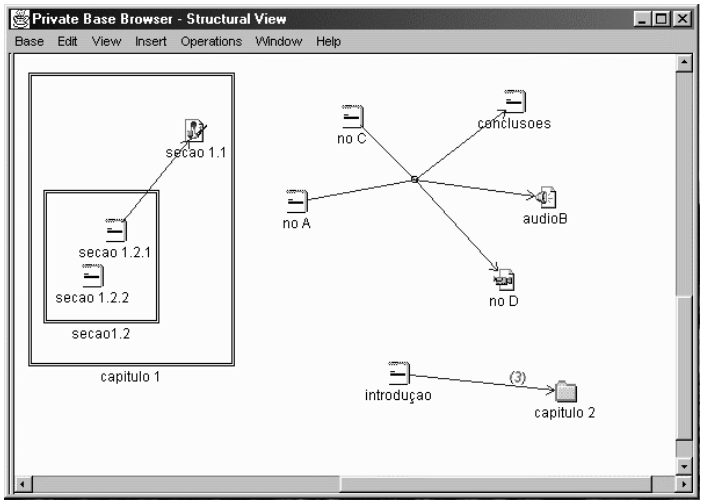


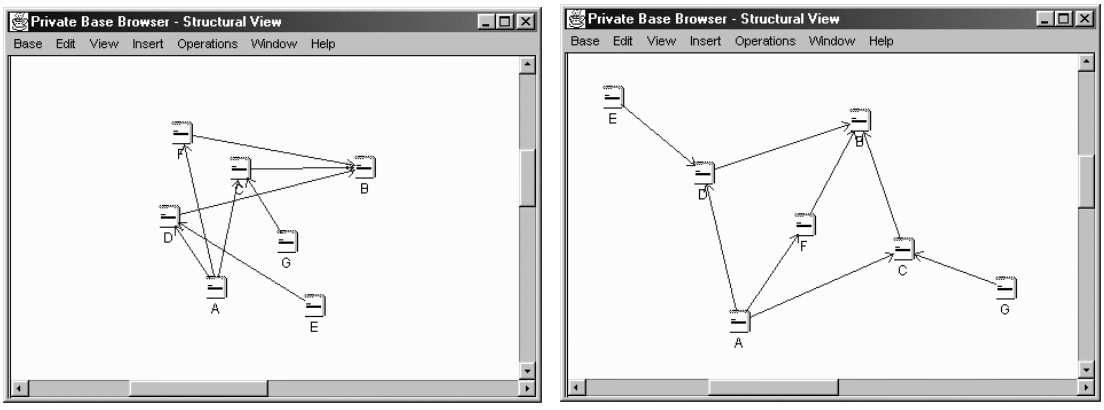
Figura 4.6 – Alterações implementadas : composições aninhadas, elos n:m e elos sobrepostos.

A Figura 4.6 exemplifica algumas das modificações que foram realizadas nas classes do VGJ, exibindo elos *n*:*m*, composições aninhadas e elos sobrepostos. Em outras palavras, a biblioteca VGJ teve de ser adaptada para o desenho de grafos compostos estendidos, conforme definido na Seção 4.1.

A interface gráfica dos navegadores do HyperProp consiste em um mapa ativo formado por um grafo composto que representa o espaço de informações hipermídia. O grafo não é um diagrama estático; seus elementos podem ser selecionados pelo usuário e movidos de acordo com sua preferência. O nó selecionado pode ter seus atributos (nome, URL, características espaciais e temporais) editados ou iniciar a apresentação do documento que representa. Quando o documento é exibido, o nó correspondente aparece em destaque no grafo, para que o usuário saiba as opções de navegação a partir deste nó.

A notação por conjuntos aninhados expressa a hierarquia dos documentos através de conteúdo espacial, com cada nó sendo posicionado dentro de seu nó pai. Este tipo de representação é bastante interessante para estruturas que definem relações hierárquicas e de referência cruzada simultaneamente, pois os relacionamentos são representados de formas distintas: por inclusão e por elos. Assim, com essa notação, obtém-se um mapa inicial com menos detalhes, pois a estrutura interna das composições não é apresentada. Para exibir seu conteúdo, o usuário deve selecionar a composição e abrí-la, navegando em profundidade por sua estrutura hierárquica. É importante destacar que, se um mesmo nó NCM aparecer mais de uma vez no grafo (no interior de várias composiçoes), ele é desenhado mais de uma vez.

A apresentação legível do grafo é fundamental para a orientação do usuário durante a navegação. Isto é obtido graças ao *layout* automático do grafo, que ocorre toda vez que a configuração deste é alterada, ou porque uma composição foi aberta (ou fechada), ou porque um nó (elo) foi inserido (removido). A Figura 4.7 mostra a importância do *layout* no desenho de um grafo.



(a) (b)

Figura 4.7 – Um mesmo grafo, desenhado aleatoriamente (a) e por meio de um algoritmo de layout (b).

O usuário do HyperProp pode forçar a reorganização do conteúdo de uma composição a qualquer momento, bastando selecionar uma opção no menu do sistema. O algoritmo é acionado apenas para o *layout* do conteúdo do nó selecionado. Se não houver nenhum nó

marcado, o *spring-embedder* atua no primeiro nível do grafo. Uma outra opção de menu pode ser selecionada para impedir que o grafo seja atualizado automaticamente. A configuração inicial é mantida (algoritmo é executado somente uma vez), e o usuário tem que dispor os elementos do grafo manualmente nas ocasiões seguintes.

4.3.1 *Spring-Embedder* Estendido

A biblioteca de desenho de grafos utilizada implementa três algoritmos diferentes para o *layout* de grafos. Um para o desenho de árvores, outro para grafos direcionados e o último para grafos não-direcionados. A Figura 4.8 ilustra um mesmo grafo desenhado por esses três algoritmos, respectivamente. Em (c), o desenho foi produzido pelo algoritmo *spring-embedder* para grafos não-direcionados genéricos, escolhido para ser utilizado pelo sistema HyperProp.

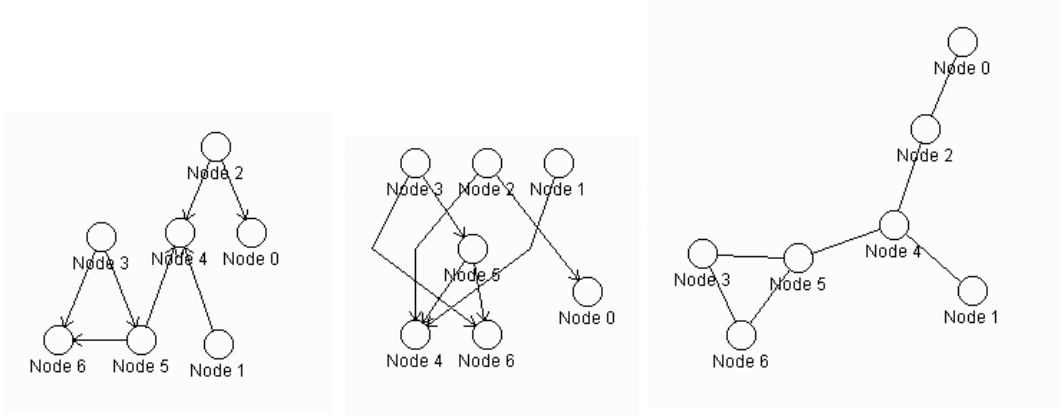


Figura 4.8 – Um mesmo grafo desenhado pelos algoritmos de layout em árvore (a), CGD (b) e spring-embedder (c).

O algoritmo para árvores dispõe em linha reta os nós de um mesmo nível, centralizando o nó pai acima dos nós filhos. Se o grafo não for uma árvore, é empregado um algoritmo de busca em largura para identificar a árvore geradora, utilizada pelo algoritmo de

layout. Se o grafo contiver ciclos, um nó que servirá de raiz deve ser escolhido pelo usuário antes do algoritmo ser aplicado.

O algoritmo implementado para grafos direcionados, também conhecido como CGD (*clan-based graph decomposition*), tem como objetivos: respeitar o sentido dos elos, de modo que o nó origem do elo fique sempre situado acima do nó destino; minimizar o cruzamento de elos e o número de quinas nos elos. O *layout* dos nós é determinado através de uma análise do grafo em subgrupos coesos e da definição de atributos que são aplicados na árvore resultante. Tais atributos, como o espaçamento entre nós, proporcionam um balanceamento tanto horizontal quanto vertical entre os nós.

O algoritmo para grafos não-direcionados é o *spring-embedder*, apresentado na seção anterior. Este algoritmo procura satisfazer alguns critérios estéticos, como criar desenhos de grafos com poucos cruzamentos de elos, manter os elos retos e com comprimento uniforme. Por ser aplicado em grafos genéricos e devido à simetria do diagrama final obtido, é o algoritmo mais adequado para o desenho de estruturas de documentos hipertexto. Entretanto, mesmo assim, foi necessário adaptá-lo para suportar as entidades do modelo NCM, como já mencionado.

Como o grafo hipertexto é originalmente direcionado, deve-se torná-lo não-direcionado antes do algoritmo ser executado. Isto é feito considerando que, para todo nó *v* adjacente a *u*, existe uma aresta contrária tal que *u* seja adjacente a *v* também. Assim, as matrizes de comprimento, menor caminho e força do *spring-embedder* são construídas de forma que o mesmo valor seja atribuído para os dois pares de nós.

Uma outra característica do algoritmo estabelece que o grafo a ser desenhado deva ser totalmente conexo. Embora desejável, não se pode garantir que um documento hipermídia tenha sempre todos os seus nós ligados por elos, portanto deve-se modificar o grafo para que

o *spring-embedder* funcione corretamente. Antes de se iniciar o algoritmo, uma rotina é executada para percorrer o grafo e identificar os grupos conexos. São criados elos invisíveis e temporários (*dummy edges*) ligando cada grupo, tornando assim o grafo totalmente conexo. Após a execução do algoritmo, esses elos criados são retirados. A Figura 4.9 mostra o resultado do *spring-embedder* estendido no *layout* de um grafo desconexo (os elos invisíveis estão representados como linhas tracejadas).

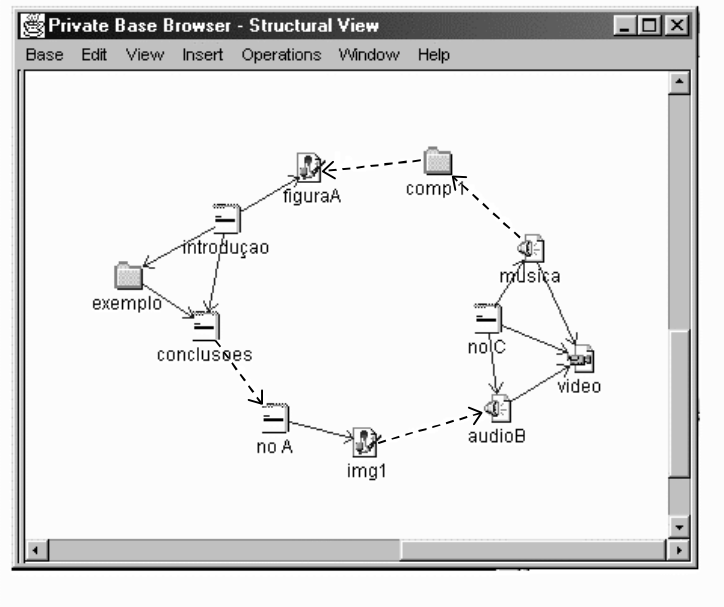
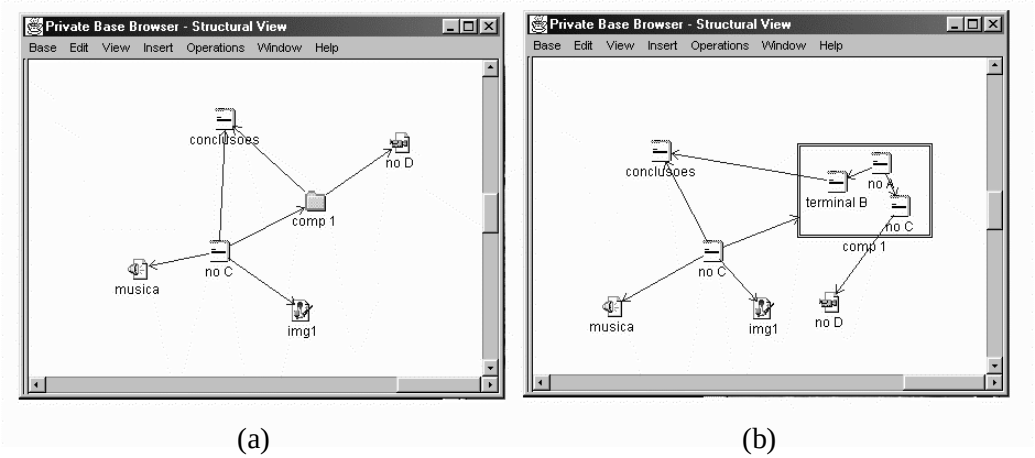


Figura 4.9 – O layout de um grafo desconexo.

O algoritmo também foi estendido para suportar grafos compostos. O conjunto de nós passados para o *spring-embedder* é limitado, para que este atue apenas em um nível do grafo de cada vez. Dessa forma, o *layout* dos subgrafos contidos nos nós compostos (contextos) é computado independentemente, dos níveis mais internos para os mais externos. Uma reorganização do conteúdo de um nó composto, causada pela abertura do próprio contexto ou pela inclusão de um novo nó, por exemplo, propaga a ativação do algoritmo de *layout* para os níveis acima, pois geralmente as dimensões do contexto redesenhado são alteradas após a execução do algoritmo. A Figura 4.10 apresenta um exemplo deste processo em um grafo composto.



(a) (b)

Figura 4.10 – A propagação no algoritmo de layout de um grafo composto.

Como parâmetro de entrada do algoritmo, é passada, além dos nós do grafo, a área disponível para o desenho, pois o diâmetro final do grafo e o comprimento das arestas são dependentes deste valor. Quando é computado o *layout* do primeiro nível do grafo, a área disponível é toda a janela do navegador; quando é desenhado o conteúdo de um contexto, esse número é calculado com base na quantidade de nós e elos presentes no contexto. Após a execução do algoritmo, uma rotina redefine o tamanho da contexto de acordo com a posição de seus nós componentes.

Foi observado que o *layout* final de grafos compostos produzidos pelo *spring-embedder* estendido, às vezes, pode não corresponder ao melhor desenho possível: podem ocorrer sobreposições de elos e nós. Isto se deve à estratégia de extensão do *spring-embedder* adotada para o *layout* de grafos compostos. Como mencionado acima, o *spring-embedder* é aplicado no interior de cada composição por vez, de forma isolada. Assim, os elos unindo nós contidos em composições diferentes não são considerados no *layout* dessas composições. Tais elos são computados nos níveis acima do grafo, mais especificamente no *layout* da primeira composição comum às perspectivas das extremidades do elo. Como o algoritmo atua

considerando apenas os componentes diretos do grafo, dependendo da configuração final do desenho, pode haver a sobreposição desses elos em nós recursivamente contidos.

A Figura 4.11 ilustra um exemplo onde acontece a sobreposição. Em (a), as composições mais internas já foram arranjadas, faltando somente o *layout* da composição *a*. O elo *<(b,x),(c,y)>*, ligando nós de composições diferentes, pode gerar sobreposições. Em (b), vemos o grafo simples equivalente, utilizado pelo *spring-embedder* para o layout de *a*. Os nós recursivamente contidos não são considerados neste grafo. Em (c), o desenho final é obtido, ocorrendo a sobreposição do elo indicado nos outros nós de *c*.

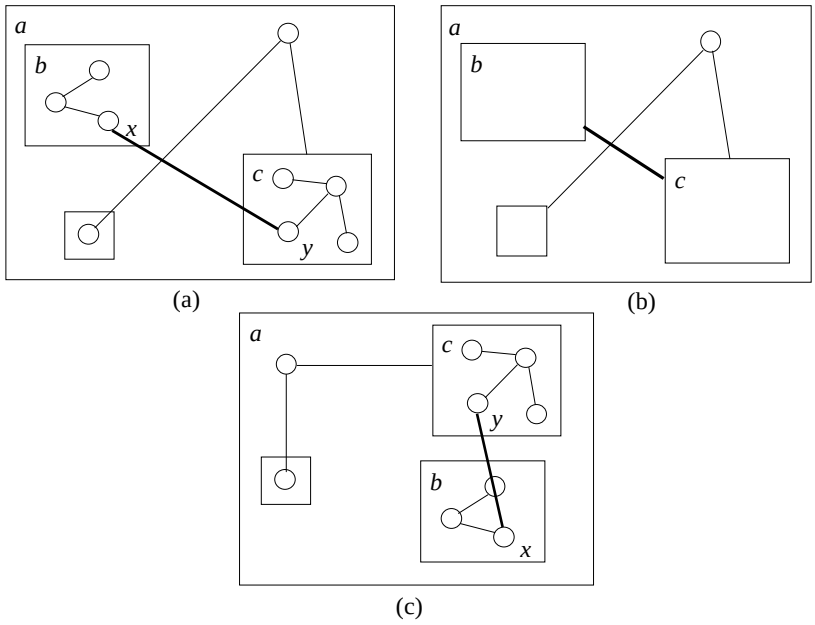


Figura 4.11 – A ocorrência de sobreposições de nós e elos no spring-embedder estendido.

Este problema não é resolvido de maneira trivial. Um procedimento que identificasse a sobreposição de elos e nós e determinasse um novo posicionamento dos nós ou a criação de "quinas" nos elos seria bastante complexo e não conpensaria seu uso. A solução talvez seria definir uma outra estratégia de extensão do *spring-embedder* para grafos compostos.

Devido ao próprio cálculo do *spring-embedder* (mencionado na seção anterior) e às rotinas de pré-processamento[10], o *layout* do grafo tende a ser um processo custoso, quando o número de nós é alto. O uso de mecanismos de filtragem, como a estratégia olho-de-peixe discutida no próximo capítulo, visa tornar o desenho mais legível, mas também contribui para contornar este problema, pois reduz o número de nós visíveis no grafo. Além disso, o fato do algoritmo ser executado em etapas num grafo composto, associadas ao conteúdo de cada composição, faz com que o esforço computacional seja menor.

4.3.2 Algoritmos Específicos de *Layout*

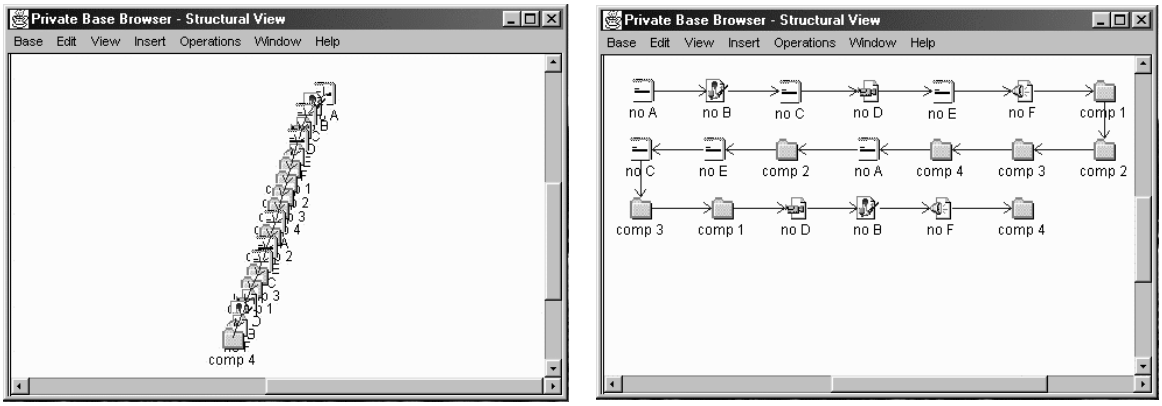
Existem dois casos particulares em que o algoritmo *spring-embedder* estendido não é a melhor opção para o desenho do grafo de entidades do modelo NCM. São eles: a exibição do conteúdo de trilhas e do primeiro nível de aninhamento das bases (base privada e hiperbase pública).

As trilhas, criadas para fornecer um histórico da navegação do usuário ou um caminho predefinido pelo autor de um documento, contêm apenas uma lista ordenada de nós, que são conectados entre si por elos que representam as relações de sucessão na trilha. Ao exibir seu conteúdo no navegador, optou-se por posicionar esses nós na ordem em que são definidos, formando uma única cadeia. Um novo tipo de *layout* teve que ser criado, uma vez que o *spring-embedder* não seria adequado para este tipo de grafo. O algoritmo tentaria desenhá-lo em uma única linha, com um resultado bastante ruim, como pode ser visto na Figura 4.12 (a). A Figura 4.12 (b) ilustra o novo *layout* para o mesmo grafo: a trilha é construída, posicionando-se o primeiro nó no canto superior esquerdo da área de desenho e

---

[10] De todas as extensões acrescentadas no *spring-embedder*, a que possui maior tempo de processamento é o algoritmo que torna o grafo totalmente conexo: $O(n^2)$.

adicionando-se os seguintes em forma de um ziguezague horizontal. Este *layout* pode ser utilizado também para exibir o conteúdo de composições seqüenciais, recentemente definidas no modelo NCM.



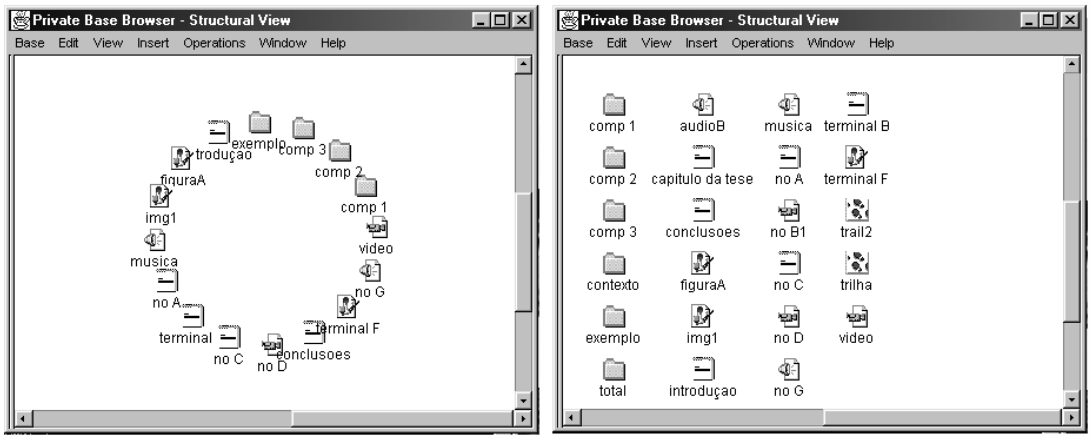
(a) (b)

Figura 4.12 – O layout de uma trilha, se for aplicado o spring-embedder estendido (a) e o algoritmo próprio (b).

Assim como as trilhas, as bases privadas e a hiperbase pública também precisam de algoritmos de *layout* específicos. Como não apresentam elos de referência[11], o *spring-embedder* torna-se inadequado para o *layout* do conteúdo destas composições. A razão é o grande número de elos que deveriam ser inseridos para tornar o grafo totalmente conexo, como determina o algoritmo. A Figura 4.13 (a) apresenta o resultado do *spring-embedder* aplicado nos nós contidos numa base privada. Como o grafo originalmente não possui elos, o novo grafo obtido forma um círculo cujo desenho torna-se esteticamente ruim, se contiver muitos nós. Para evitar este tipo de desenho, desenvolveu-se um algoritmo próprio para as bases privadas e a hiperbase pública, que ordena alfabeticamente seus nós em colunas, separando as composições do nós terminais (Figura 4.13 (b)).

---

[11] O modelo NCM permite a criação de elos entre nós e nós de anotação [47]. Porém, o baixo número destes elos em relação ao número de nós não justifica o uso do *spring-embedder*.



(a) (b)

Figura 4.13 – O layout de uma base privada, se for aplicado o spring-embedder estendido (a) e o algoritmo próprio (b).

É importante frisar que este algoritmo apenas é aplicado no primeiro nível das bases, devido à falta de elos. O *layout* das composições contidas nas bases é computado normalmente pelo *spring-embedder*. Entretanto, o artifício de dispor os nós em colunas também pode ser usado em composições que possuem um número bastante reduzido de elos em relação ao número de nós.

Capítulo 5

Visões Olho-de-Peixe para Documentos Hipermídia

No nosso dia a dia, normalmente conhecemos o que está ao nosso redor com grande detalhe, enquanto que identificamos apenas alguns pontos de referência a maiores distâncias. Uma pessoa sabe os nomes das ruas de sua vizinhança, os nomes das cidades próximas a sua e os nomes das cidades mais importantes dos outros estados. Caso trabalhe ou tenha parentes em uma outra cidade, eventualmente a conhecerá com detalhes também. Em outras palavras, existem duas coisas que definem a percepção de um objeto: sua distância e a importância *a priori* para o observador. Este conceito foi primeiramente formalizado por Furnas [18], ao propor a visão olho-de-peixe (*fisheye view*) como estratégia de filtragem, sendo mais tarde estendido ao desenho de grafos por Sarkar e Brown [45].

Neste capítulo, a proposta original de Furnas para a visão olho-de-peixe é introduzida e a extensão apresentada em [33] para grafos compostos analisada. Este estudo leva em consideração tanto os relacionamentos de referência quanto os de inclusão entre documentos

para o cálculo da distância requerida pelo olho-de-peixe. Também são descritos os detalhes de implementação da visão olho-de-peixe realizadas no sistema HyperProp.

5.1 Visões com Olho-de-Peixe

Um problema conhecido enfrentado pelos usuários de um sistema hipermídia é a desorientação ao navegar pelos documentos hipertextos. Esta desorientação é causada por uma característica inerente ao conjunto de hiperdocumentos: ser composto por um número arbitrariamente elevado de nós e de elos [10]. É cada vez mais comum em navegadores, a presença de ferramentas gráficas que forneçam uma visão global resumida do espaço de informações para orientar o usuário durante a navegação. À medida que o usuário navega pelo documento, visualizando o conteúdo dos nós, o sistema atualiza um mapa do espaço hipermídia que exibe a posição corrente do usuário na estrutura global.

Além de orientar a navegação, a exibição do espaço hipermídia por meio destes mapas pode auxiliar na localização de informações. Navegando pela estrutura de um documento, o usuário pode consultar mais facilmente um determinado dado, sem o custo de percorrer o conteúdo dos outros nós do documento.

Dentre as técnicas mais comuns para a apresentação de visões do espaço de informações, podem ser citadas a exibição de todo o grafo e o uso de recursos de *zoom* e *scroll* em partes do grafo. Entretanto, a técnica que se mostra mais eficaz, mas também mais complexa, é a filtragem de nós pouco relevantes para o usuário [45], principalmente em modelos estruturados, como o NCM. A dificuldade é identificar quais nós realmente interessam ao usuário.

A visão olho-de-peixe proposta por Furnas age como uma lente, preservando os detalhes próximos a um ponto escolhido e, à medida que se afasta deste ponto, exibindo menos informação, mostrando apenas as referências mais importantes. Nos mapas do espaço hipermídia, a filtragem baseada na estratégia olho-de-peixe fornece, em uma única visão, os detalhes locais referentes à posição do usuário durante o processo de navegação (nó em foco), sem perder a noção da estrutura global dos documentos. Assim, ao mesmo tempo em que é mantida uma noção geral da posição do nó corrente na estrutura hierárquica, obtém-se a informação das possibilidades de navegação a partir deste nó.

Como vimos no início deste capítulo, a distância de um nó em relação ao observador e sua importância *a priori* são os fatores preponderantes para determinar se este nó deve ou não ser exibido. A estratégia básica do olho-de-peixe [18] se utiliza destes dois fatores para definir uma função de grau de interesse (*degree of interest*), que atribui a cada nó do espaço hipermídia um valor, representando o grau de interesse do usuário em relação ao nó em foco. A idéia principal da visão olho-de-peixe é que esta função (*DOI*) aumente com a importância *a priori* (*API*) e diminua com a distância (*D*):

$$DOI(x, y) = API(x) - D(x, y),$$

onde *DOI* é o grau de interesse do usuário para o nó *x* em relação ao nó em foco corrente *y*.

A filtragem dos nós é realizada, escolhendo um determinado valor *K* para o corte, de modo a exibir somente os nós *x* que possuírem um $DOI(x,y) \geq K$. Variando o valor de *K*, que pode ser interpretado como o nível de detalhe desejado, obtém-se diferentes visões olho-de-peixe para o mesmo grafo.

5.2 Extensão da Estratégia Olho-de-Peixe para Grafos Compostos

A proposta original de Furnas é aplicada em estruturas onde a função *DOI* pode ser facilmente definida, como em listas e árvores. No caso de documentos hipertexto, embora a estrutura das composições aninhadas forme uma árvore, os elos referenciais forçam uma representação em grafo. Em [45], são propostas algumas extensões à visão olho-de-peixe para o caso de grafos genéricos, como a alteração do tamanho dos nós e de sua posição para destacar a área próxima ao foco. Outros trabalhos [39] estendem o cálculo da função *DOI* para grafos compostos, mas só a distância hierárquica (relações de inclusão) é usada para computar a distância entre nós.

A ferramenta de filtragem utilizada no sistema HyperProp para a construção dos mapas é baseada em uma extensão da estratégia olho-de-peixe para grafos compostos [33], que leva em conta tanto as relações de inclusão quanto as relações de adjacência para computar os nós e elos visíveis. Desta forma, para calcular a função grau de interesse para cada nó, são considerados os dois modos de navegação em grafos compostos: seguindo os elos de referência e percorrendo em profundidade a estrutura de contextos.

Para grafos compostos, a importância *a priori API(x)* de um nó *x* é dada pelo valor negativo do nível de aninhamento do nó dentro da perspectiva observada em relação ao contexto mais externo. Para um nó x_m na perspectiva $P = (x_0, x_1, ..., x_m)$, a $API(x_i)$, com $0 \leq i \leq m$, é dada por:

$$API(x_i) = -i$$

A Figura 5.1 apresenta um exemplo de um grafo composto. A importância *a priori* do nó *g* é igual a -3.

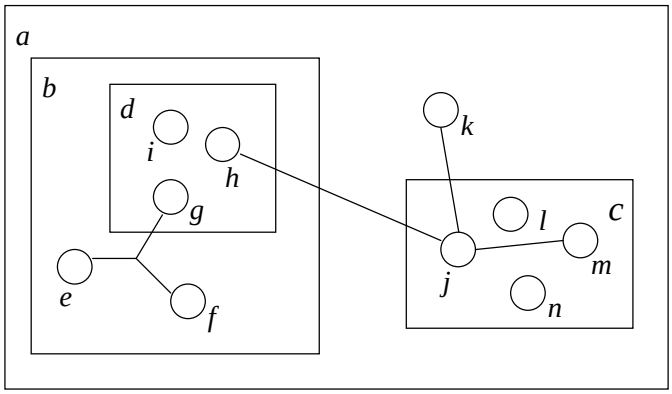


Figura 5.1 – Diagrama de um grafo composto.

A segunda componente da função *DOI*, a distância *D* entre um nó *x* e o foco *y*, é dada pela seguinte função:

$$D(x, y) = \min\left(D_c(x, y), D_e(x, y)\right),$$

onde $D_c(x,y)$ é a distância mínima entre *x* e *y* considerando somente a navegação em profundidade pelas perspectivas, e $D_e(x,y)$ é a distância mínima entre *x* e *y* considerando somente a navegação por elos. Se não há um caminho de elos ligando os dois nós, a distância $D(x,y)$ é equivalente a $D_c(x,y)$.

A distância considerando a navegação em profundidade $D_c(x,y)$, do foco *y* para o nó *x* é dada por:

$$D_c(x, y) = dista + distd$$

onde *dista* é a distância ascendente, calculada adicionando-se uma unidade para cada passo no caminho ascendente entre *y* e *x*, e *distd* é a distância descendente, calculada adicionando-se uma unidade para cada passo no caminho descendente entre *y* e *x* e, ao final, somando-se o valor $(dista + w_c)$[12]. O ponto de virada entre o caminho ascendente e descendente é o primeiro nó ancestral comum aos nós *x* e *y*. Caso a distância entre os dois nós

[12] A definição dos pesos w_c e w_e será dada posteriormente nesta seção.

não possua a componente ascendente ou descendente, fato que ocorre respectivamente quando o foco pertence à perspectiva do nó ou vice-versa, a distância correspondente é igual a zero.

No exemplo da Figura 5.1, a distância considerando a navegação por profundidade entre os nós *m* (foco) e *b*, respectivamente pertencentes às perspectivas $P_1 = (a,c,m)$ e $P_2 = (a,b)$, é igual a $D_e(m,b) = 5$, considerando $w_c = 0$. Os passos da distância ascendente são *c* e *a* (*dista = 2*) e da distância descendente é *b* (*distd = 3*).

Para calcular a distância considerando a navegação por elos $D_e(x,y)$, deve-se, para cada caminho de elos conectando *x* e *y*, adicionar uma unidade a cada segmento de elo do caminho. Entende-se por segmento de elo, o trecho do elo entre dois cruzamentos consecutivos com nós do grafo composto. A distância $D_e(x,y)$ será igual ao valor mínimo encontrado entre todos os caminhos entre *x* e *y*. Note que, para a definição do caminho, não são consideradas as direções dos elos, mesmo para grafos direcionados. Portanto, $D_e(x,y) = D_e(y,x)$.

Dado um elo $<(f_0, ..., f_k), (v_0, ..., v_n)>$, com $k \geq 0$ e $n \geq 0$, ligando os nós f_k e v_n, sendo o nó f_k pertencente à perspectiva $P_1 = (c_0, ..., c_m, f_0, ..., f_k)$ e nós v_i pertencentes à perspectiva $P_2 = (c_0, ..., c_m, v_0, ..., v_i)$ para $i \in [0, n]$, a distância $D_e(f_k, v_i)$ é dada pela expressão:

$$D_e(f_k, v_i) = k + i + 1 + w_e \quad [12]$$

No caso do elo estar ancorado na própria composição que o contém, o cálculo da distância é diferente. Seja o elo $<(f_0, ..., f_k), (c_m)>$, com $k \geq 0$ e $m \geq 0$, ligando os nós f_k e c_m, sendo o nó f_k pertencente à perspectiva $P_1 = (c_0, ..., c_m, f_0, ..., f_k)$ e o nó c_m pertencente à perspectiva $P_2 = (c_0, ..., c_m)$, a distância $D_e(f_k, c_m)$ é dada por:

$$D_e(f_k, c_m) = k + 1 + w_e$$

Para um elo multiponto-a-multiponto (*n*:*m*), que conecta mais de dois nós, o cálculo da distância $D_e(x,y)$ é feito entre todas as extremidades do elo, como se houvesse vários elos

simples conectando-as. Por exemplo, a Figura 5.2 mostra um elo *n*:*m* interligando quatro nós (*a*, *b*, *c* e *d*). O valor da distância D_e do nó *a* para todos os outros nós do grafo é calculado por meio de três falsos elos.

Figura 5.2 – Cálculo da distância D_e para elos n:m.

Voltando à Figura 5.1, a distância considerando a navegação por elos entre os nós *h* e *m*, por exemplo, é $D_e(h,m) = 5$, porque no caminho entre os dois nós existe um elo *<(m),(j)>* e outro elo *<(c,j),(b,d,h)>*, que apresenta 4 segmentos: (*j*,*c*), (*c*,*b*), (*b*,*d*) e (*d*,*h*). O valor da distância D_e depende de como os elos são definidos no grafo. Se, neste exemplo, o elo entre os nós *m* e *j* fosse definido como <(c,m),(c,j)>, a distância $D_e(h,m)$ seria 7, pois haveria mais dois segmentos no caminho.

Ainda na Figura 5.1, podemos fazer o cálculo completo da função *DOI* para o nó *h*, considerando o nó *j* como o foco:

$$API(h) = -3$$
$$D_c(h, j) = 2 + (3 + 2) = 7$$
$$D_e(h, j) = 4$$
$$D(h, j) = \min(7,4) = 4$$
$$DOI(h, j) = API(h) - D(h, j) = -3 - 4 = -7$$

O algoritmo responsável por calcular a visão olho-de-peixe utilizado no HyperProp, satisfaz algumas propriedades para melhorar a orientação do usuário ao navegar pelo espaço hipermídia [33]:

i) Se um nó é exibido no diagrama, todos os nós que compõem sua perspectiva são sempre exibidos, garantindo que a noção de contextos não seja perdida;

ii) Todos os nós diretamente ligados ao nó em foco são sempre exibidos, informando ao usuário as possibilidades de navegação a partir do foco;

iii) Se um nó de contexto é o nó em foco, seus componentes são sempre exibidos;

iv) Nós pertencentes a perspectivas que não sejam a do nó em foco têm prioridade decrescente de exibição do contexto mais externo ao mais interno. Dada a perspectiva $P = (c_0, c_1, ..., c_k)$, os nós $c_0, c_1, ..., c_k$ são apresentados nesta ordem de prioridade;

v) Nós que não são conectados ao nó em foco e que são componentes de contextos pertencentes à perspectiva do foco, têm prioridade decrescente de exibição do contexto mais interno ao mais externo. Por exemplo, supondo que x seja o nó em foco e que pertença a perspectiva $P = (c_0, c_1, ..., c_k, x)$, e que o contexto c_i contenha os nós n_i, para $i \in [0, k]$. Então, os nós $n_k, n_{k-1}, ..., n_0$ terão esta ordem de prioridade de apresentação.

A propriedade i) é sempre satisfeita, pois o valor acrescentado em D_c é compensado pela menor *API* do nó, fazendo com que $DOI(x,y) = DOI(y,y) = API(foco)$ para qualquer nó x pertencente à perspectiva do foco y. Para atender ii), é preciso limitar o valor máximo do nível de corte K. Este valor deve ser igual a $API(foco) - 1$, que corresponde ao número de segmentos de elo até alcançar o nó base da perspectiva da outra extremidade do elo, somada a *API* deste nó. A terceira propriedade também limita o valor máximo de K. Para exibir os componentes do foco, é preciso adicionar uma unidade para a API e outra para a Dc destes

nós, ou seja, K deve ser limitado por API*(foco) – 2*. As propriedades iv) e v) são sempre satisfeitas, devido às próprias características da função DOI,

Deve ser ressaltado que, para o algoritmo atender estas propriedades e exibir um número mínimo de nós que ajude à orientação do usuário, o limite máximo do valor de corte *K* da visão olho-de-peixe deve ser igual a *API(foco) – 2*, se o nó em foco for um contexto, e *API(foco) – 1*, se o foco for um nó terminal.

As constantes w_c e w_e que aparecem nas fórmulas das distâncias em profundidade e por elos representam os pesos que estes dois métodos de navegação têm no cálculo da visão olho-de-peixe. A diferença entre eles determina uma prioridade na exibição dos nós relacionados ao nó em foco pela estrutura hierárquica ou por elos. Estes pesos, juntamente com o limite *K*, são parâmetros que podem ser ajustados pelo usuário para alterar a visão do espaço hipermídia.

5.3 Implementação

A visão olho-de-peixe está disponível nos navegadores de base privada, de hiperbase pública e de base de contextos de versões do HyperProp. A filtragem é automaticamente preparada assim que um navegador exibe seu mapa hipertexto, mas pode ser desativada ou ativada novamente, se o usuário desejar, através de uma opção de menu. Toda vez que é iniciado, o mecanismo de olho-de-peixe força o fechamento de todas as composições abertas no diagrama de documentos, bastando o usuário focalizar um nó no mapa, para que a filtragem dos nós seja realizada. Pelo menu do navegador, o usuário também pode designar um nó

selecionado como referência (nós sempre visíveis) e alterar os parâmetros de configuração da filtragem olho-de-peixe.

A estratégia da visão olho-de-peixe proposta por Furnas pode ser facilmente aplicada em listas, árvores e outras estruturas hierárquicas. Para estes casos, o cálculo atinge um tempo de processamento de ordem logarítmica, sendo a função *DOI* facilmente recalculada quando o foco é mudado [18]. No caso de documentos hipertexto, representados por grafos compostos, a possível existência de mais de um caminho entre os dois nós e a falta de um nó raiz dificultam o cálculo da visão olho-de-peixe. Assim, a função que computa a distância considerando a navegação por elos tende a apresentar um tempo de processamento muito elevado, quando o grafo possui uma alta conectividade. Para aplicações que precisam manter uma interatividade com o usuário, como é o caso dos navegadores, é importante reduzir ao máximo este esforço de processamento.

Na implementação da visão olho-de-peixe no HyperProp, esta preocupação foi um requisito do projeto. O algoritmo utilizado na primeira versão do sistema [32] foi revisto e, principalmente o cálculo da distância em elos, melhorado para tentar reduzir seu tempo de processamento. As maiores diferenças foram obtidas devido à utilização da mesma estrutura de dados presente na biblioteca de *layout*[13]. Neste modelo, os nós possuem como atributos vetores de nós componentes e listas de adjacências, que ajudam a identificar mais facilmente as relações de inclusão e referência do grafo composto. Como esta biblioteca foi estendida para suportar elos *n*:*m*, pôde-se implementar o cálculo da visão olho-de-peixe para este tipo de elo.

[13] O sistema D-ABDUCTOR, empregado na primeira versão do sistema, possuía uma estrutura interna bastante complexa. Para o cálculo da visão olho-de-peixe, portanto, criou-se uma estrutura de dados paralela.

Além disso, para calcular a distância em elos dos nós em relação ao nó em foco, considera-se não apenas os elos contextuais deste, mas todos os seus elos visíveis[14].

Uma outra maneira de acelerar o processamento da visão olho-de-peixe é limitar o número de nós que têm sua função *DOI* calculada. Apenas são considerados os nós que são visitados e os componentes diretos de uma composição, quando esta é aberta. Assim, ao se iniciar uma sessão de trabalho, somente para os nós do primeiro nível de aninhamento da base privada (ou da hiperbase pública) é computado o grau de interesse. À medida que o usuário navega pelos documentos, são incluídos os nós visitados, garantindo que apenas aqueles nós que realmente interessam ao usuário sejam exibidos. Este ainda pode explicitamente escolher quais nós entram ou saem do cálculo da visão olho-de-peixe, bastando abrir ou fechar manualmente a composição que os contém.

Além disso, o cálculo da função *DOI* só é ser refeito quando o nó em foco mudar (afetando a referência das distâncias obtidas) ou quando houver alterações na estrutura do documento (a inclusão/remoção de elos ou de contextos influi no valor da importância *a priori* e das distância entre nós).

Quando o usuário navega pelo documento, alterando o nó em o foco, pode ser que algumas composições se fechem e outras sejam abertas automaticamente. Isto vai depender da função *DOI* de seus componentes em relação ao novo foco. A característica, oferecida pela interface gráfica do navegador, de poder representar o conteúdo de uma composição apenas por um nó auxilia a manter a legibilidade do mapa de documentos. Entretanto, as alterações na visibilidade de nós e elos, ao calcular o grau de interesse para o novo foco, podem reduzir a eficiência da ferramenta de navegação, pois a mudança abrupta do mapa normalmente destrói a

---

[14] Elos que ancoram em um nó, definidos em qualquer contexto da perspectiva do nó, são chamados *elos contextuais*. Ao conjunto de elos que ancoram ou passam por um nó, dá-se o nome de *elos visíveis*.

visão mental do usuário, causando desorientação. Este problema pode ser parcialmente resolvido se forem utilizadas técnicas de animação ao mostrar as mudanças no grafo. Este recurso, embora torne mais lento o processo de atualização da visão, reduz a mudança instantânea da configuração do grafo, preservando o mapa mental do usuário.

A biblioteca de desenho de grafos usada na implementação do sistema foi alterada para dar a noção de animação durante o reposicionamento de nós e a ativação de composições. No primeiro caso, os nós são desenhados nas posições parciais encontradas pelo algoritmo de *layout*. No caso da abertura (ou fechamento) de composições, calcula-se uma interpolação entre suas dimensões inicial e final, e atualiza-se o desenho da composição com alguns deste valores intermediários.

Devido às técnicas de *layout* automático empregadas na construção do grafo, a posição de um nó na tela não é fixa, o que também dificulta a orientação do usuário durante a navegação. Para reduzir este problema, alguns nós podem ser definidos como pontos de referência (*landmarks*), sendo sempre exibidos no diagrama, como forma de ajudar o usuário a manter seu senso de localização. Os nós de referência podem ser utilizados também durante a edição de um documento, forçando a visualização de um determinado nó.

Cada usuário pode definir seu conjunto de nós de referência, que são exibidos de forma diferenciada no diagrama, para facilitar sua identificação (aparecem em negrito, como mostra a Figura 5.3). No algoritmo da visão olho-de-peixe, os nós *x* escolhidos como referência não precisam ter sua função grau de interesse calculada, bastando assegurar que seu valor satisfaça à condição $DOI(x,y) \geq K$. Isto é garantido sempre, fazendo para todo *landmark* x, $DOI(x,y) = K$.

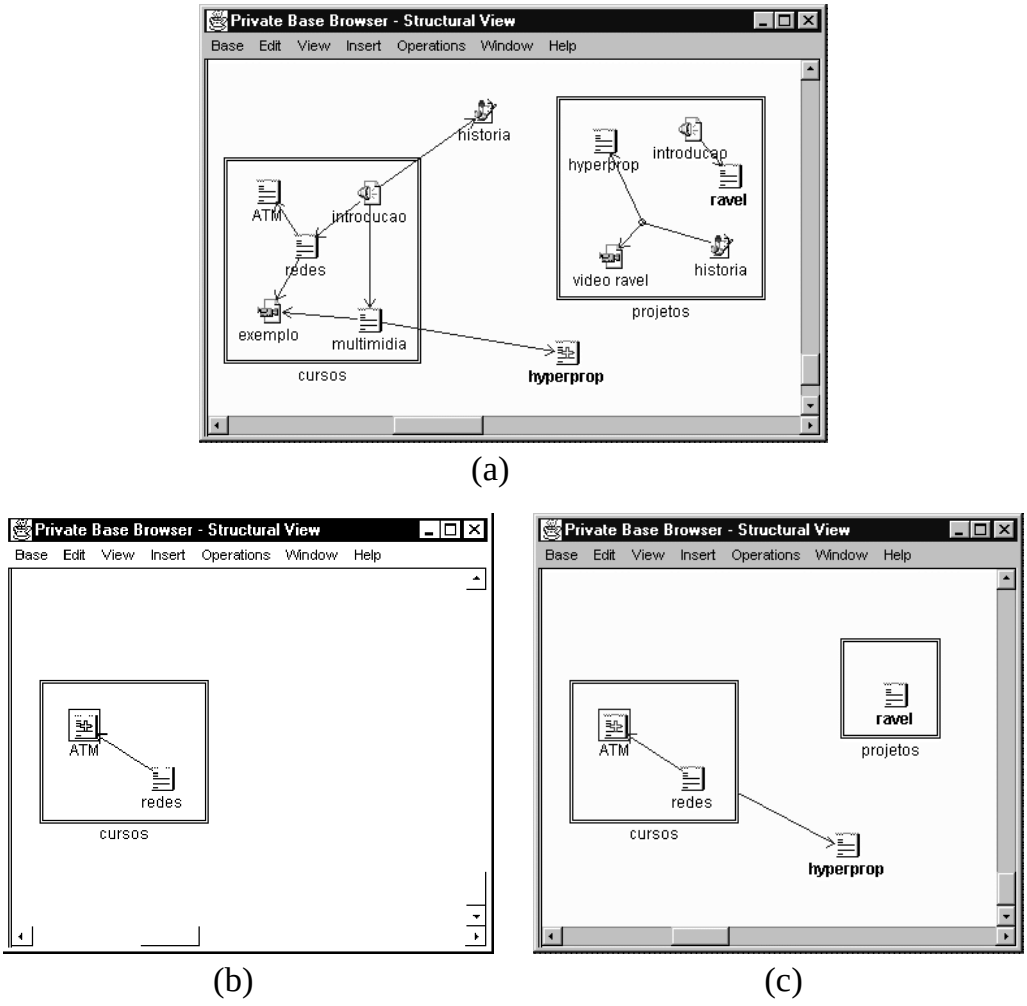


Figura 5.3 – Utilização dos nós de referência: em (a) o mapa de documentos sem filtragem; em (b), a visão do mesmo mapa com olho-de-peixe somente e em (c), o resultado do olho-de-peixe com dois nós de referência.

A Figura 5.3 ilustra o uso dos nós de referência com a visão olho-de-peixe. Em (a), vê-se a estrutura de todo o documento, sem a aplicação do algoritmo de filtragem. Em (b), já com o olho-de-peixe ativado, tem-se o nó *ATM* como foco. Nota-se que os nós de referência *hyperprop* e *ravel* são forçados a serem exibidos para auxiliar a navegação.

Uma característica que deve estar presente em qualquer ferramenta de navegação é a sua capacidade de poder ser configurada pelo usuário, de acordo com suas necessidades e objetivos. No HyperProp, o mecanismo que implementa a visão olho-de-peixe conta com três

parâmetros para sua configuração: o valor de corte da função *DOI* e os pesos dados às formas de navegação hipertexto.

No olho-de-peixe, o valor de corte (*threshold*) *K* determina a filtragem dos nós: apenas os nós que possuem um grau de interesse acima deste valor são exibidos no mapa. O valor de *K* pode ser configurado pelo usuário durante a navegação, obtendo visões do grafo de documentos com maior ou menor número de nós, de acordo com o nível de detalhe desejado. É importante ressaltar que quando o nível de detalhe é modificado, a atualização do mapa é realizada diretamente. Como o valor da função *DOI* para cada nó fica guardado em memória, não há necessidade de um novo processamento para recalcular as distâncias, apenas o teste final ($DOI(x,y) \geq K$) é realizado novamente.

Os valores válidos para *K* (aqueles que efetivamente alteram a visão estrutural do grafo) são definidos quando o foco é escolhido e a função *DOI* calculada. Todos os valores possíveis de *DOI* encontrados para os nós do grafo são reservados e disponibilizados para o usuário através de uma interface para configuração, que pode ser vista na Figura 5.4. Por esta interface, o usuário pode alterar o corte *K* escolhendo um valor percentual que representa o nível de detalhe da informação apresentada no mapa. O valor máximo para *K* significa 0% de detalhe (mostrando o menor número de nós possível); o detalhe máximo (100%) corresponde ao valor mínimo para o corte, quando todos os nós são exibidos. Somente algumas percentagens estão disponíveis, justamente aquelas que representam os valores válidos para *K*.

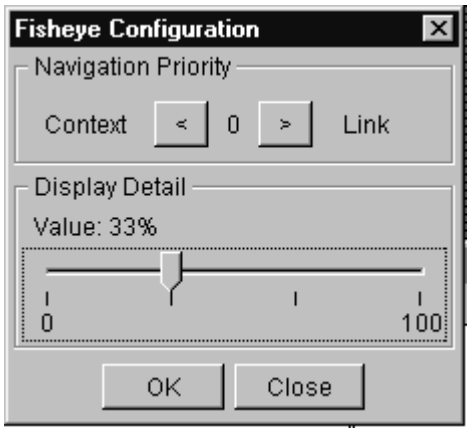


Figura 5.4 – Interface para configuração da filtragem olho-de-peixe.

Nesta mesma interface de configuração (Figura 5.4), o usuário pode também escolher qual prioridade atribuir às formas de navegação hipertexto (por profundidade ou por elos) na visão olho-de-peixe. Este valor expressa a diferença entre os pesos w_c e w_e, que aparecem nas fórmulas das distâncias D_c e D_e, respectivamente. Para efeito de cálculo, quando deseja-se dar prioridade à distância em elos, por exemplo, w_e é tornado nulo e o valor absoluto determinado pelo usuário[15] (w_c) é somado a D_c. No caso da distância em contextos, é feito o oposto.

Da mesma forma que na escolha do grau de detalhe, as distâncias D_c e D_e não precisam ser recalculadas quando este parâmetro é alterado, pois, para encontrar o novo *DOI*, é necessário determinar somente o valor mínimo entre D_c e D_e, considerando os novos valores dos pesos:

$$D(x, y) = \min\left(D_c(x, y) + w_c, D_e(x, y) + w_e\right)$$

Assim, se for dada prioridade à exibição dos nós adjacentes ao nó em foco, o valor final da distância em contextos cresce, tendendo a função *DOI* a considerar D_e.

[15] Na implementação da interface de configuração dos pesos, o usuário atribui valores negativos para priorizar D_c e valores positivos para dar preferência a D_e. Esta maneira foi escolhida apenas para auxiliar o entendimento do processo, mas w_c e w_e assumem sempre valores positivos.

Analogamente, se o usuário der prioridade aos nós relacionados ao foco por inclusões, a distância em elos aumenta e D_c passa a ser considerada.

Como os parâmetros w_c e w_e produzem um deslocamento (*bias*) no cálculo de *DOI*, o valor máximo de *K* também deve ser ajustado, para que as propriedades do algoritmo do olho-de-peixe continuem a ser satisfeitas. Assim, o valor máximo para *K* deve ser igual a *API(foco) – 1 – w*, se o foco for um nó terminal, e *API(foco) – 2 – w*, se o nó em foco for um contexto, onde *w* é o maior valor entre w_c e w_e.

As alterações nos parâmetros de configuração permanecem ativas durante a sessão do usuário, sendo válidas para a exibição da estrutura de qualquer documento, até que um novo valor seja informado. Para ilustrar as modificações que ocorrem num diagrama de documentos, quando os parâmetros de configuração são manipulados pelo usuário, um exemplo de um documento é apresentado nas figuras seguintes.

A Figura 5.5 mostra o navegador de base privada exibindo a estrutura completa de um documento, que contém cinco composições (*comp1*, *comp2*, *comp3*, *comp4* e *comp5*) e três nós terminais (*no B*, *no D* e *no H*). Na Figura 5.6, é apresentada a visão inicial do mesmo documento, quando ativada a filtragem olho-de-peixe. Nota-se que apenas seus componentes diretos (os nós do primeiro nível de aninhamento) são mostrados. A partir desta visão, o usuário pode navegar em profundidade pela estrutura do documento, focalizando um determinado nó para revelar seu conteúdo. Se focalizar o nó *comp4*, por exemplo, a visão que se obtém é apresentada na Figura 5.7 (a). Para efeito de comparação, o mesmo grafo, sem a filtragem, é mostrado em Figura 5.7 (b). Se o usuário então escolher *comp2*, a estrutura visível do documento passa a ser outra, como mostra a Figura 5.8 (a). E, na Figura 5.8 (b), a mesma visão sem o olho-de-peixe. Nos dois casos, como o grau de detalhe da filtragem é o

menor possível, apenas os nós obrigatórios (nós diretamente ligados ao foco ou o conteúdo destes) aparecem nos desenhos.

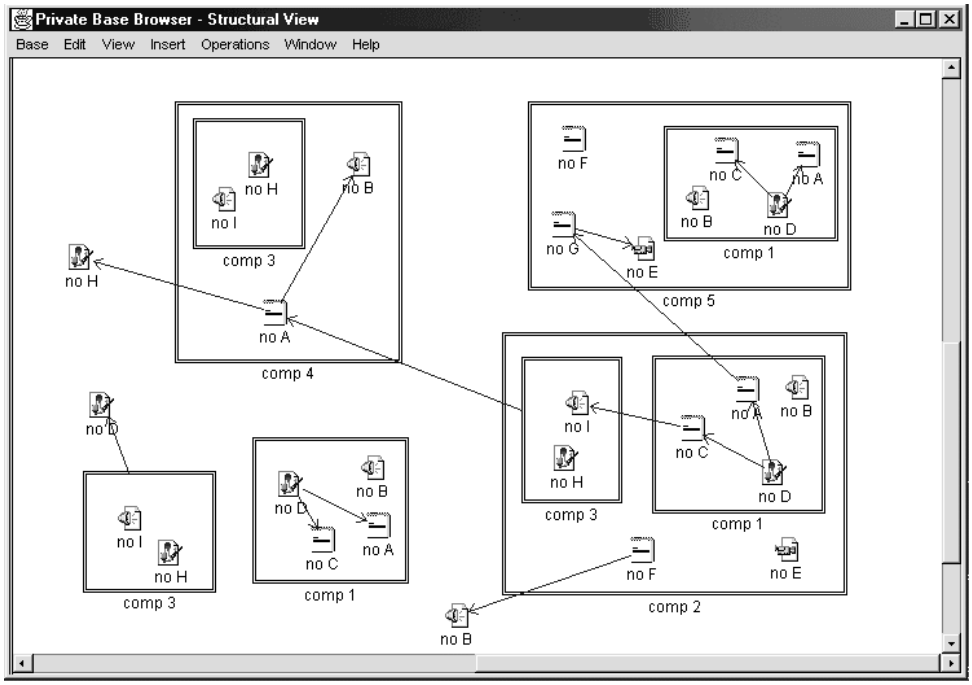


Figura 5.5 – A estrutura completa de um documento hipermídia.

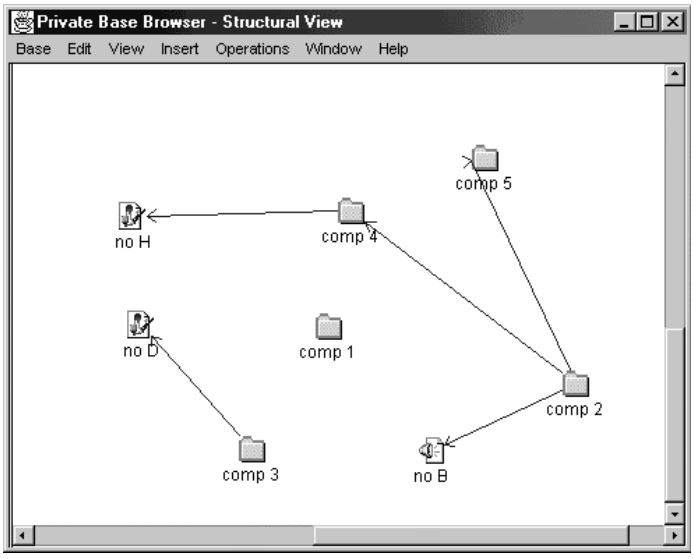
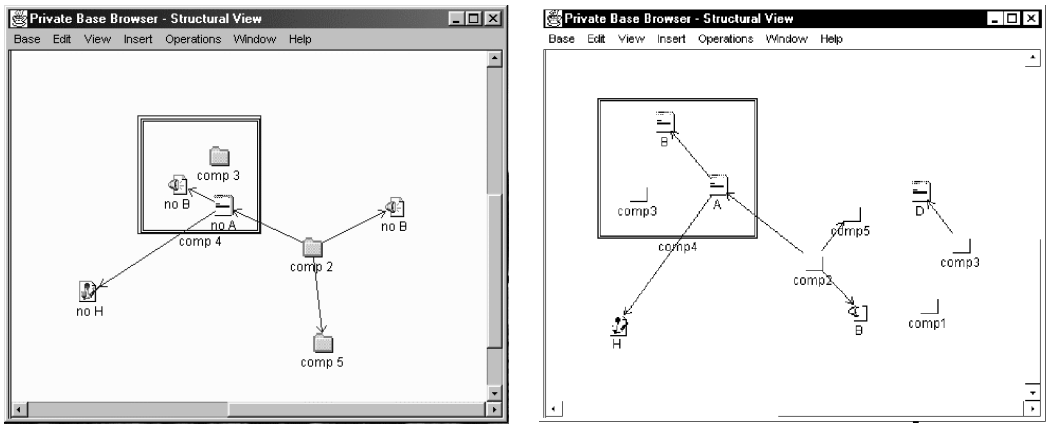
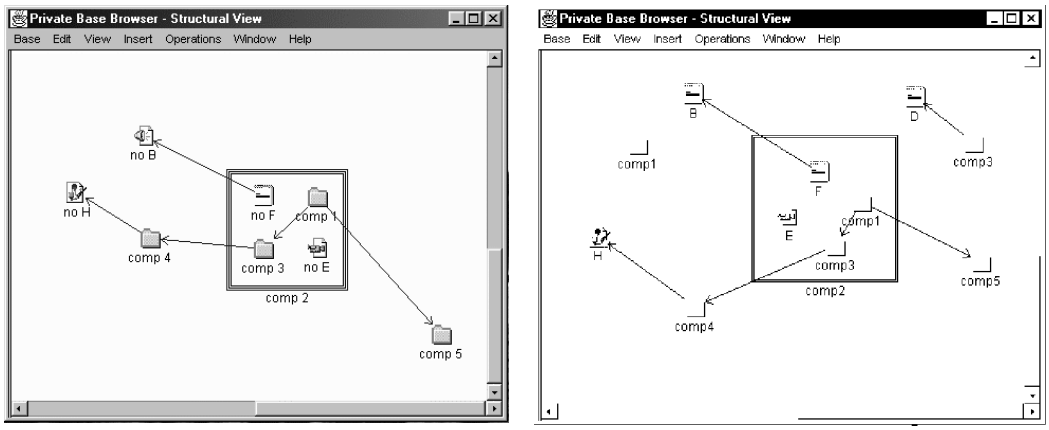


Figura 5.6 – A visão inicial do documento.



(a) (b)

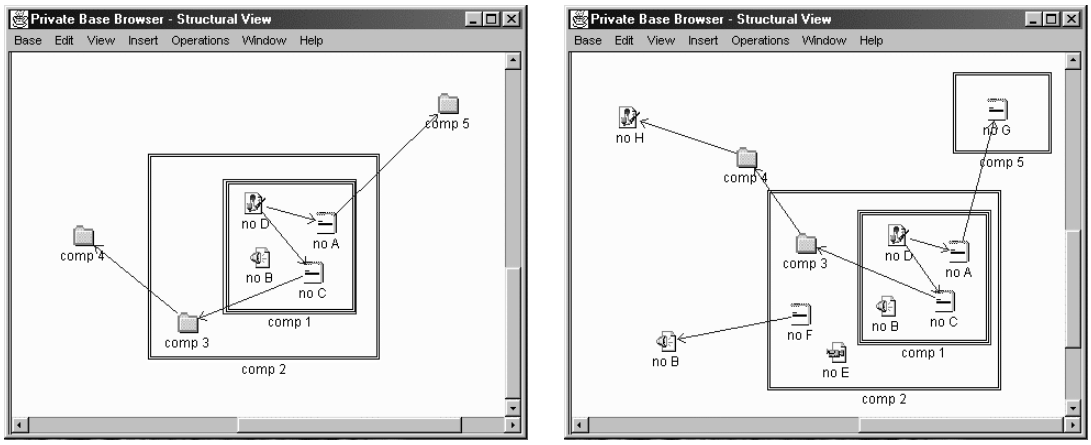
Figura 5.7 – Focalizando o nó comp4, com a filtragem olho-de-peixe (a) e sem a filtragem (b).



(a) (b)

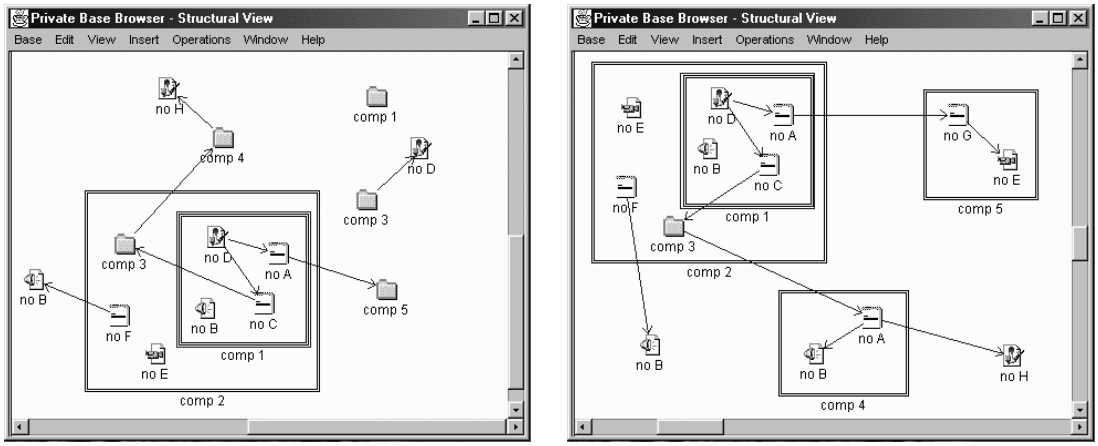
Figura 5.8 – Continuando a navegação pelo documento: foco em comp2, com o olho-de-peixe (a) e sem a filtragem (b).

Focalizando o nó *comp1*, pertencente à composição *comp2*, obtém-se a estrutura exibida na Figura 5.9 (a). Mantendo o foco em *comp1*, mas aumentando o grau de detalhe do desenho (de 0% para 33%), tem-se a Figura 5.9 (b). Para o mesmo nó em foco, se for alterado o parâmetro de configuração da visão olho-de-peixe que expressa a prioridade de exibição dos nós, primeiro para mostrar os nós com menor distância em contextos do foco e depois para priorizar os nós relacionados ao foco por elos, obtém-se grafos diferentes. As Figura 5.10 (a) e (b) representam as duas situações, respectivamente.



(a) (b)

Figura 5.9 – Focalizando o nó comp1 (componente de comp2) (a). comp1 ainda é o nó em foco, mas a visão foi reconfigurada para apresentar mais detalhes (b).



(a) (b)

Figura 5.10 – Focalizando o mesmo nó comp1 (mantendo o nível de detalhe), primeiro priorizando a distância em contextos (a) e depois a distância em elos (b).

Capítulo 6

Autoria de Estruturas de Documentos Hipermídia

Além de oferecer ferramentas para orientar a navegação no espaço de informações, um dos principais objetivos no projeto do HyperProp é permitir a criação e edição de documentos hipermídia. Dependendo de seus direitos de acesso, um usuário pode atuar como autor e publicar remotamente seus documentos por meio do próprio sistema. O papel do autor é um passo além da característica dos navegadores usuais adotados, por exemplo, pela WWW, onde seus usuários podem apenas consultar a informação disponível.

Este capítulo trata particularmente sobre as facilidades na autoria da estrutura de documentos hipermídia oferecidas pelo HyperProp. Assim, são apresentados os detalhes de implementação dos navegadores que compõem o módulo cliente do sistema, incluindo seus diagramas de classes e uma descrição de todas as suas funcionalidades.

6.1 Navegadores do Sistema HyperProp

Como foi visto na Seção 3.3, o sistema HyperProp oferece vários navegadores gráficos para auxiliar o usuário na localização, autoria e apresentação de documentos hipermídia. A interface gráfica destes navegadores consiste num mapa formado por um grafo, cujos nós representam os fragmentos de informação dos documentos e os elos, as relações entre eles. Dependendo do tipo do navegador, este grafo pode ser composto, com nós contendo subgrafos, para expressar a organização lógica dos documentos. No HyperProp, os navegadores de grafos compostos são utilizados para exibir a estrutura da hiperbase pública, da base de contextos de versões e da base privada.

Os navegadores de grafos simples mostram apenas um nível de aninhamento do conteúdo de composições, sendo empregados na exibição de contextos de usuário, de contextos de versões e de contextos de variantes. Como os mapas destes navegadores contêm um número menor de nós, não são aplicados mecanismos de filtragem de informação. Portanto, a técnica da visão olho-de-peixe, descrita no Capítulo 5, é adotada apenas nos mapas de grafos compostos.

Os navegadores de grafos compostos podem ainda oferecer uma visão de como os nós de contexto estão organizados hierarquicamente. Essa visão exibe o aninhamento dos contextos por meio de uma estrutura em árvore, similar à árvore de diretórios encontrada, por exemplo, no Front Page. O usuário pode abrir ou fechar ramos da árvore e, ao selecionar um contexto, o conteúdo deste é mostrado na visão em grafo composto. A árvore facilita a localização de um determinado nó, particularmente no navegador de hiperbase pública, como se verá, por ter uma navegação mais rápida e direta. Além disso, limita um escopo para os dados exibidos no grafo, servindo como um mecanismo natural de filtragem. A Figura 6.1

mostra a tela do navegador de hiperbase pública exibindo o conteúdo da composição *comp2*, selecionada na visão em árvore.

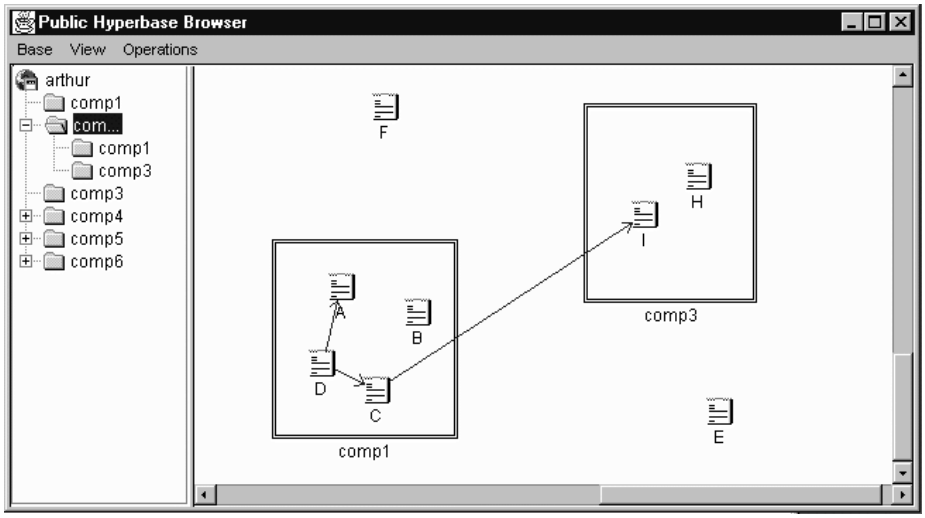


Figura 6.1 – A visão em árvore do navegador de grafo composto.

O navegador de hiperbase pública exibe a estrutura dos objetos de armazenamento existentes no servidor HyperProp, orientando o usuário na busca dos documentos desejados. O navegador é utilizado principalmente para a localização e transferência de documentos, não possuindo assim, funções de autoria nem de apresentação de documentos. Uma vez encontrados, os documentos selecionados são transferidos para a base privada por meio das operações de versionamento (*Check-out* e *Open*) para que possam ser manipulados localmente pelo usuário. Caso este queira salvar seu trabalho – para que qualquer outro usuário do sistema possa referenciar seus documentos, por exemplo –, ele deve utilizar a função *Check-in* para guardar na hiperbase pública as novas versões de nós criadas.

A base privada se constitui na área de trabalho do usuário, onde ele pode navegar, editar e acionar a apresentação de documentos. A edição da estrutura dos documentos somente é permitida no navegador de base privada, onde são manipuladas as versões objetos de dados dos nós trazidos da hiperbase pública. As operações de autoria de documentos

incluem a criação/remoção de nós terminais e de contextos, a criação/remoção de elos, com a possibilidade de adição de mútiplos pontos terminais (no caso de elos *n:m*), além da criação e edição de âncoras, descritores e trilhas.

6.1.1 Ambientes de Apresentação e Autoria

A partir do navegador de base privada, o usuário pode iniciar o ambiente do sistema para a apresentação de documentos, que aciona exibidores específicos para a visualização do conteúdo de cada tipo de mídia envolvida. A apresentação de nós contexto do usuário é realizada pelo exibidor de contextos, que mostra um mapa com o conteúdo das composições, permitindo ao usuário navegar em profundidade pelos nós do documento. Da mesma forma que o navegador de hiperbase pública, o navegador de contextos não apresenta uma interface para autoria da estrutura de documentos.

No ambiente de apresentação da base privada, as marcações de sincronismo temporal e espacial entre os nós, definidas pelos descritores e elos, estão ativadas. No ambiente de autoria, ao contrário, estas marcações ficam desabilitadas, pois é importante que todos os nós fiquem disponíveis para o usuário compor o documento final. Similarmente, quando algum documento é exibido, as ferramentas de autoria são inibidas para evitar que o usuário possa editá-lo, gerando inconsistências entre os dois ambientes.

Alguns recursos são empregados para melhorar a orientação do usuário, ao alternar entre os dois ambientes. Por exemplo, quando um nó é apresentado, seu ícone do navegador de base privada é posto em destaque e, à medida que o usuário navega pelo documento, a visão olho-de-peixe da base privada é atualizada para focalizar o nó corrente. Quando o usuário passa o cursor sobre uma âncora do nó que está sendo apresentado num exibidor, o elo correspondente a esta âncora no mapa da base privada também é destacado, indicando o caminho a ser seguido.

Atualmente, os dois ambientes são implementados de maneira isolada: uma alteração num nó no ambiente de autoria implica na recriação de sua instância correspondente no ambiente de apresentação, na próxima vez em que é acionada. Em um trabalho futuro, seria interessante desenvolver um controle que identificasse as edições realizadas no documento e o atualizasse automaticamente no ambiente de apresentação, como faz o Kaomi [24], possibilitando, assim, a compilação incremental da exibição dos documentos.

No NCM, a hiperbase pública tem uma base de contextos de versões associada, de forma que todas as versões de dados derivadas dos nós de armazenamento ou de dados são inseridas automaticamente nos contextos de versões que contêm estes nós OA, contextos de versões que, por sua vez, estão contidos na base de contextos de versões associada.

Disponíveis no sistema, os navegadores da base de contextos de versões, de contexto de versões e de contexto de variantes são utilizados para a visualização e autoria do conteúdo dos respectivos contextos. Os dois primeiros navegadores permitem que o histórico de versionamento de documentos, que expressa as relações de derivação entre nós, possa ser editado explicitamente pelo usuário. Eles fazem parte do controle de versões do HyperProp, tema da dissertação desenvolvida em [36], e não serão analisados no presente trabalho. O navegador de contexto de variantes é usado na autoria dos nós no plano de representação.

6.1.2 Autoria de Nós Objeto de Representação

O plano de representação contém os nós objetos de representação (OR), formados pela associação de um nó objeto de dados (OD) e um descritor, como mencionado na Seção 3.1. Um nó OR possui métodos para exibir e editar seu conteúdo, inexistentes nos nós OD. No ambiente de apresentação, o conteúdo dos nós OR é visualizado por exibidores específicos (configurados pelo usuário) para cada tipo de nó. No caso de nós de contexto, utiliza-se o exibidor de contexto de usuário, apresentado na seção anterior.

No ambiente de autoria, contudo, o exibidor de contextos não seria o mais adequado para tratar os nós OR, pois teria que passar a mostrar todos os nós de representação contidos no contexto escolhido. Neste exibidor, somente os nós componentes cujos descritores estão associados à composição pai aparecem no mapa, uma vez que a apresentação iniciada a partir da seleção de um nó no mapa representa a navegação em profundidade nessa composição. Outra desvantagem é o escopo limitado dos nós OR apresentados: apenas aqueles referentes à perspectiva selecionada são mostrados no mapa do exibidor de contextos. Seria importante poder reunir todos os nós OR derivados de um mesmo nó OD.

A edição de nós OR também não poderia ser realizada pelo navegador de base privada, próprio para a autoria. Como este navegador é usado basicamente para editar a estrutura do documento, os nós exibidos são considerados objetos de dados.

Outros fatores contribuíram para a escolha de se criar um navegador exclusivo para a edição de nós objetos de representação, em vez da apresentação de uma OR-base privada. Como um mesmo nó OD pode gerar vários nós OR, o mapa da base privada teria mais nós, o que certamente prejudicaria a orientação do usuário. Além disso, alguns problemas práticos relativos à edição de nós OR foram identificados. Por exemplo, a opção de criar um nó OR equivaleria a efetivamente criar um nó (OD) ou associar um nó OD existente a um novo descritor? No mapa, como um elo entre dois nós OR teria seu descritor associado modificado, sem precisar remover e recriar o elo? Ao focalizar um nó OR na visão olho-de-peixe, como assegurar que os outros nós OR derivados do mesmo nó OD, na mesma perspectiva, estariam sempre visíveis? Foi necessário, portanto, desenvolver um novo navegador para a autoria de nós OR: o navegador de contexto de variantes.

O navegador de contexto de variantes é acionado a partir do navegador de base privada, ao ser selecionado um determinado nó no mapa e ativada a opção de menu

correspondente. É exibido então, em uma pequena janela, um grafo com os nós OR derivados do nó OD selecionado. O usuário pode escolher entre exibir todos os nós OR ou apenas aqueles relativos à perspectiva selecionada. A seleção dos nós terminais neste grafo ativa programas específicos (processadores de texto, *softwares* gráficos, etc.) configurados pelo usuário, para a edição do conteúdo do nó, de acordo com o tipo de mídia envolvida.

Embora tratando-se de nós OR, quando são exibidos para edição, suas âncoras de sincronismo espacial e temporal não são habilitadas, só aparecendo na apresentação do documento. A noção de tempo de projeto e tempo de execução se aplica perfeitamente aos ambientes de autoria e apresentação do sistema. Com o objetivo de orientar o usuário, podem ser desenvolvidos mecanismos que relacionam os nós entre os navegadores de contexto de variantes e de base privada, como foi feito entre este e o exibidor de contextos. Seria interessante, por exemplo, ao selecionar uma variante, indicar no mapa da base privada de qual elemento é proveniente o descritor que a compõe: do elo, da composição que contém o nó ou do próprio nó.

O projeto do HyperProp prevê ainda o desenvolvimento de uma interface que possibilite a edição do comportamento temporal dos documentos NCM. Um novo navegador mostraria os elementos de um documento alinhados no tempo e permitiria a edição gráfica de suas relações de sincronismo. Entretanto, a edição temporal, assim como a edição do sincronismo espacial entre os elementos e o suporte à edição do conteúdo dos nós terminais, são trabalhos futuros.

A Tabela 6.1 resume as principais características dos navegadores do sistema HyperProp. Para cada um deles, são listados o tipo de grafo que exibem, a qual ambiente do sistema pertencem, se possuem o mecanismo de filtragem e o tipo de objetos que são manipulados.

Navegadores	Características			
	Grafo	**Ambiente**	**Olho-de-peixe**	**Objetos**
Hiperbase Pública	Composto	---	Sim	Armazenamento
Base Privada	Composto	Autoria	Sim	Dados
Contexto de Usuário	Simples	Apresentação	Não	Representação
Contexto de Versões	Simples	Autoria	Não	Dados
Base Contexto de Versões	Composto	Autoria	Sim	Dados
Contexto de Variantes	Simples	Autoria	Não	Representação

Tabela 6.1 – As características dos navegadores do HyperProp.

6.2 Implementação

Os navegadores fazem parte do módulo cliente do HyperProp, cuja arquitetura é mostrada na Figura 3.4. O projeto dos navegadores, por sua vez, pode ser dividido em cinco submódulos: *VGJ_Graph*, *Browser*, *PrivateBaseBrowser*, *PublicHyperbaseBrowser* e *ContextBrowser*, como pode ser visto na Figura 6.2.

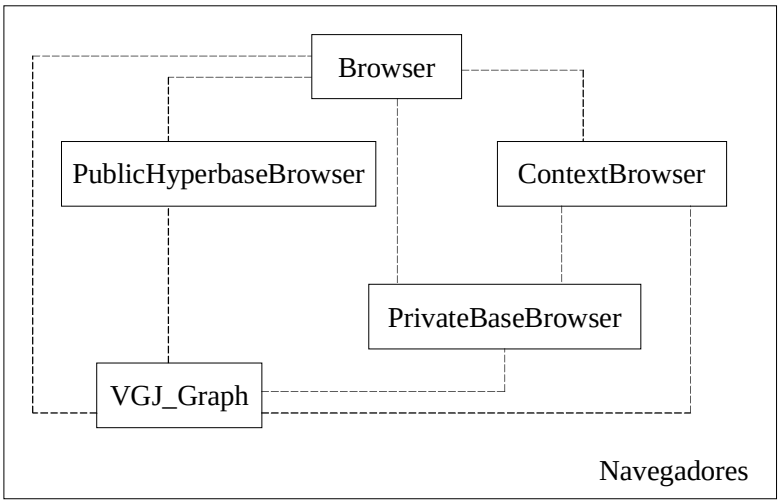


Figura 6.2 – Estrutura modular das classes dos navegadores do HyperProp.

Conforme mencionado na Seção 4.3, várias modificações foram realizadas na biblioteca de desenho de grafos, para dar suporte à estrutura de grafos exigida pelo modelo NCM, principalmente no que diz respeito às composições aninhadas e aos elos *n*:*m*. Essas classes compõem o submódulo *VGJ_Graph*, cujo diagrama é apresentado na Figura 6.3. As classes *GraphCanvas* e *GraphPanel* determinam a área para o desenho do grafo e tratam os eventos (*mouse*, teclado, etc.) gerados pelo usuário. A estrutura de dados do grafo é definida pelas classes *Graph*, *GraphTool*, *Node*, *NodeList* e *Edge*. O algoritmo de *layout* é calculado pela classe *NCMSpring*.

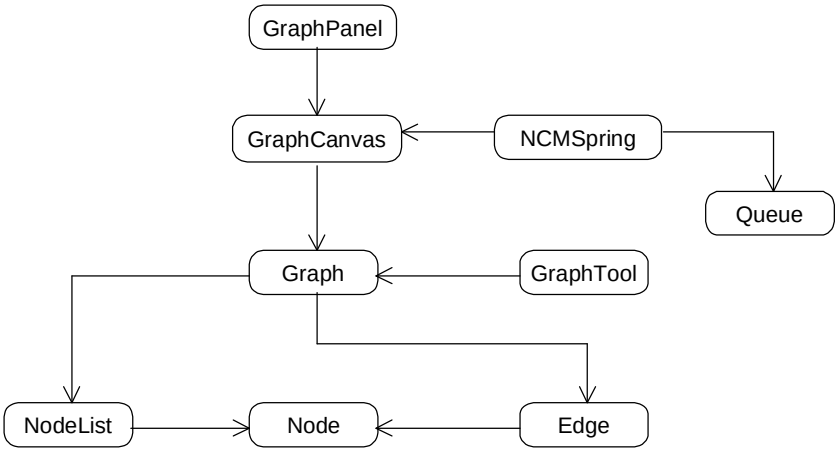


Figura 6.3 – Diagrama de classes do submódulo VGJ_Graph.

O submódulo *Browser* abrange a hierarquia das classes comuns a todos os navegadores. São classes que contêm os elementos gráficos básicos dos navegadores, como o mapa de grafos e a visão em árvore; e suas funções principais, como a visualização do grafo (com o *layout* automático) e a filtragem olho-de-peixe. Essas classes são especializadas para cada tipo de navegador, compondo os outros submódulos do projeto. A Figura 6.4 mostra o diagrama de classes do submódulo *Browser*.

As classes *TreeGraphBrowserWindow* e *GraphBrowserWindow* são derivadas da classe abstrata *BrowserWindow*, e correspondem respectivamente aos navegadores de grafo composto e de grafo simples. A classe *FisheyeBrowserWindow*, descendente de *TreeGraphBrowserWindow*, inclui o suporte à visão olho-de-peixe. As classes *TreeToolHyperProp*, *GraphToolHyperProp* e *FisheyeGraphToolHyperProp* reúnem os métodos que manipulam os elementos gráficos das visões dos respectivos navegadores, fazendo a ligação destes elementos com os objetos NCM.

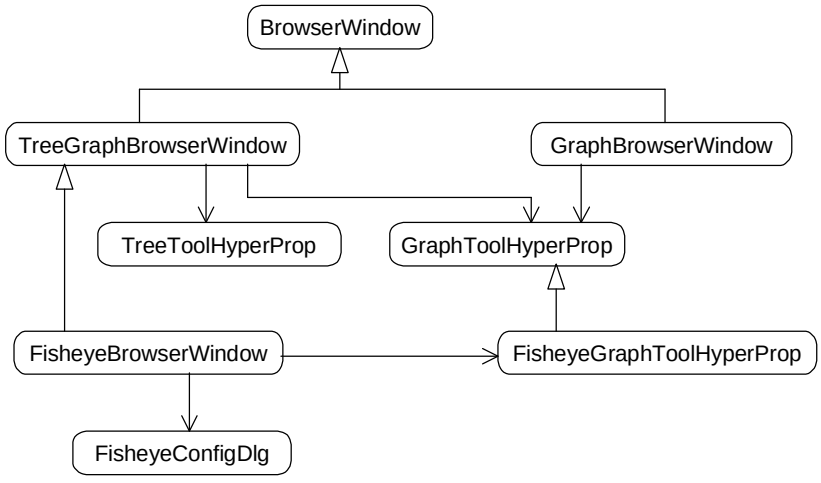


Figura 6.4 – Diagrama de classes do submódulo Browser.

No submódulo *PrivateBaseBrowser* (Figura 6.5), a classe *PrivateBaseBrowserWindow*, uma especialização de *FisheyeBrowserWindow*, implementa a

janela do navegador de base privada, cujas funções de edição de grafo estão presentes na classe *GraphToolPrivateBase*. Esta classe, por sua vez, herda de *FisheyeGraphToolHyperProp* as demais funções. As outras classes representam as caixas de diálogo para a criação e edição dos componentes de documentos hipermídia.

Analogamente, no submódulo *PublicHyperbaseBrowser*, existem as classes *PublicHyperbaseBrowserWindow* e *GraphToolPublicHyperbase*, que formam o navegador de hiperbase pública. Para o submódulo *ContextBrowser*, existem as classes *ContextBrowserWindow* (descendente de *GraphBrowserWindow*) e *GraphToolContext*, que implementam o exibidor de contextos. A classe *AuthoringObservable* define a interface entre os ambiente de apresentação e de autoria, sinalizando, para o navegador de base privada, determinadas ações ocorridas no exibidor de contextos. As Figuras 6.6 (a) e (b) apresentam os diagramas de classes dos submódulos *PublicHyperbaseBrowser* e *ContextBrowser*, respectivamente.

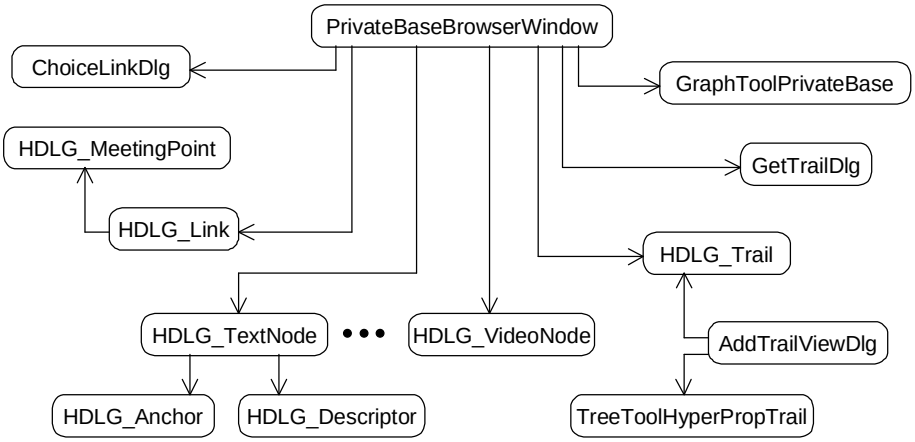


Figura 6.5 – Diagrama de classes do submódulo PrivateBaseBrowser.

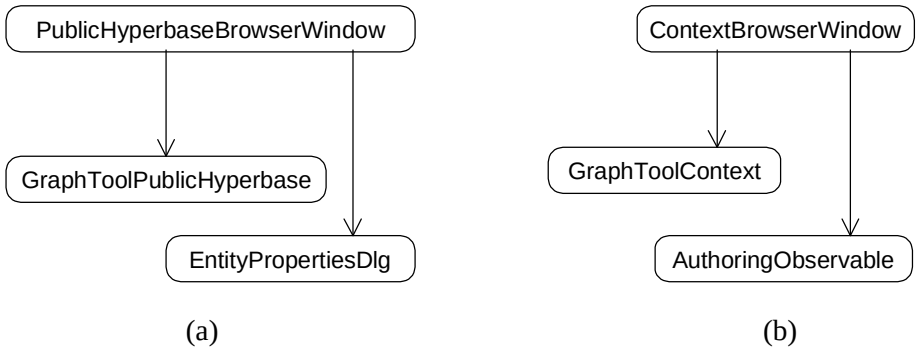


Figura 6.6 – Diagramas de classes dos submódulos PublicHyperbaseBrowser (a) e ContextBrowser (b).

Nas seções seguintes, são apresentadas as funcionalidades que foram implementadas na ferramenta de autoria e navegação do sistema HyperProp. São descritas todas as opções de menu dos navegadores que compõem os submódulos *PrivateBaseBrowser*, *PublicHyperbaseBrowser* e *ContextBrowser*, focalizando as funções de edição de estruturas de documentos.

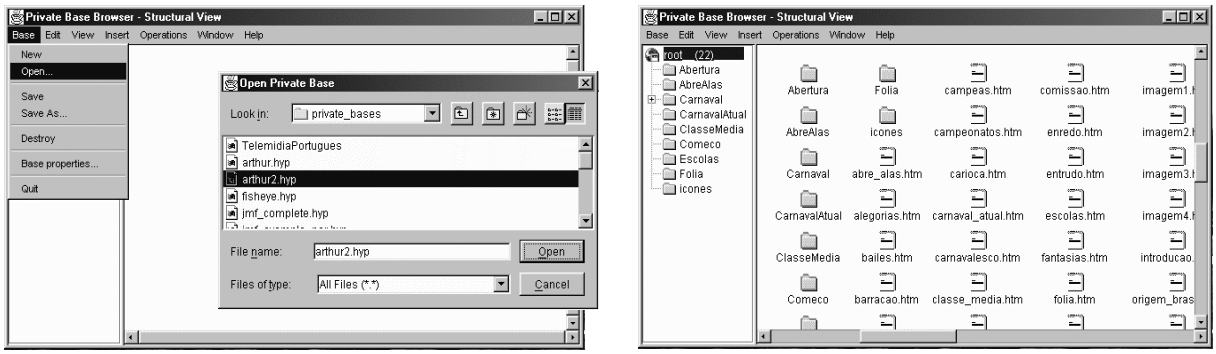
6.2.1 Navegador de Base Privada

Os clientes HyperProp iniciam uma sessão de trabalho conectando-se ao servidor NCM através de uma porta TCP comum. Por esta conexão, o usuário tem acesso a todas as informações contidas na hiperbase pública mantida pelo servidor. Após estabelecer a conexão, o navegador de base privada é iniciado no cliente, primeiramente vazio. Caso o usuário queira consultar um documento existente no servidor, pode abrir o navegador de hiperbase pública, encontrar o nó desejado e arrastá-lo para a base privada através das operações de versionamento. Se for criar um novo nó, basta utilizar as ferramentas de autoria do navegador de base privada, não sendo necessário abrir o navegador da hiperbase.

A barra de menus do navegador de base privada é composta por funções de *base*, *visualização*, *edição*, *operação* e *janela*. Algumas funções da barra de menus também são acessíveis pelo menu *popup*, quando este é acionado pelo clique do botão direito do *mouse*

sobre um nó no mapa estrutural. Essas funções são específicas para o nó selecionado, como ativar o algoritmo de *layout*, abrir ou fechar composição, focalizar o nó, mostrar suas propriedades ou ainda ativar a apresentação do nó.

As funções de base atuam sobre toda a base privada, podendo o usuário criar uma nova base (***new***), limpar totalmente seu conteúdo (***destroy***), salvar a base em disco (***save, save as***) e recuperá-la novamente (***open***). A opção por guardar localmente a base privada em arquivo é recomendada para o caso, por exemplo, em que os documentos editados ainda não estão prontos para serem disponibilizados. Posteriormente, o usuário pode retomar seu trabalho, carregando os dados guardados de sua sessão. A Figura 6.7 mostra o navegador abrindo uma base privada a partir de um arquivo já existente.



(a) (b)

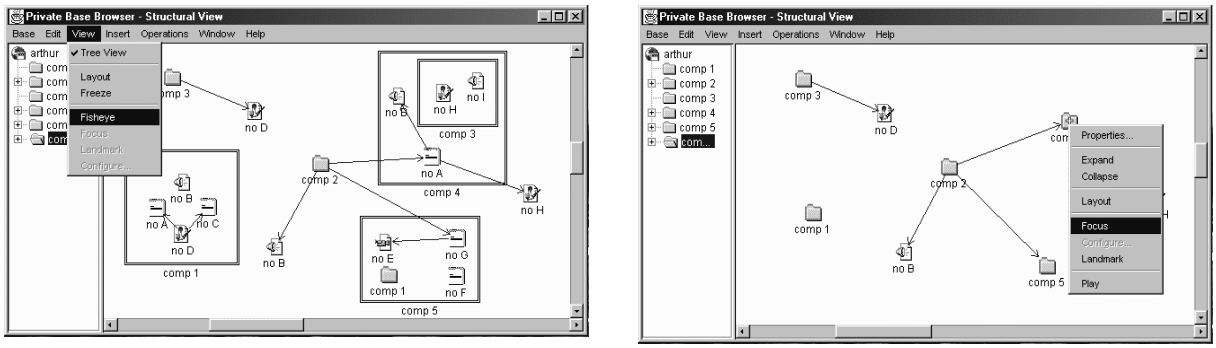
Figura 6.7 – O navegador, inicialmente vazio, lê de um arquivo (a) a base privada do usuário (b).

O navegador de base privada oferece também a possibilidade de ler (***import***) e gravar (***export***) documentos em arquivos NCL, escritos na linguagem declarativa criada para representar os documentos NCM [4]. Com o intuito de promover o intercâmbio entre documentos de sistemas hipermídia diferentes, o HyperProp suporta também a leitura e a gravação de documentos em arquivos SMIL [42].

As funções de visualização compreendem basicamente a ativação dos algoritmos de *layout* e da visão olho-de-peixe. O *spring-embedder* é aplicado automaticamente quando a configuração do mapa de documentos é alterada, mas, por uma opção do menu (***layout***), o usuário pode explicitamente acionar o algoritmo de *layout* no primeiro nível do grafo ou, se uma composição estiver selecionada no mapa, apenas no interior desta composição. Se a opção (***freeze***) for marcada, o *layout* automático é inibido. Neste caso, este só será executado por opção de menu, ou na primeira vez em que uma composição for aberta. Nas aberturas seguintes ou quando o conteúdo da composição for modificado (pela inclusão ou remoção de nós), a disposição dos nós não será alterada.

A visão olho-de-peixe deve ser ativada (***fisheye***), para que as demais funções do algoritmo fiquem habilitadas. Neste momento, todas as composições do mapa são fechadas, para que apenas os nós do primeiro nível do grafo sejam incluídos no cálculo da função grau de interesse. O usuário pode adicionar outros nós no mapa, abrindo composições ou focalizando um determinado nó através da opção de menu (***focus***)[16]. Quando um nó é focalizado, a filtragem olho-de-peixe é realizada e os nós menos relacionados com o foco somem do mapa. A Figura 6.8 ilustra a ativação da visão olho-de-peixe pelo menu (a) e, já com todas as composições fechadas, o início da navegação pelo documento, com o usuário focalizando um nó pelo menu *popup* (b).

[16] Conforme visto na seção 5.2, ao focalizar-se um nó de contexto, seu nós componentes automaticamente tornam-se visíveis devido à limitação do valor máximo do corte da visão olho-de-peixe.



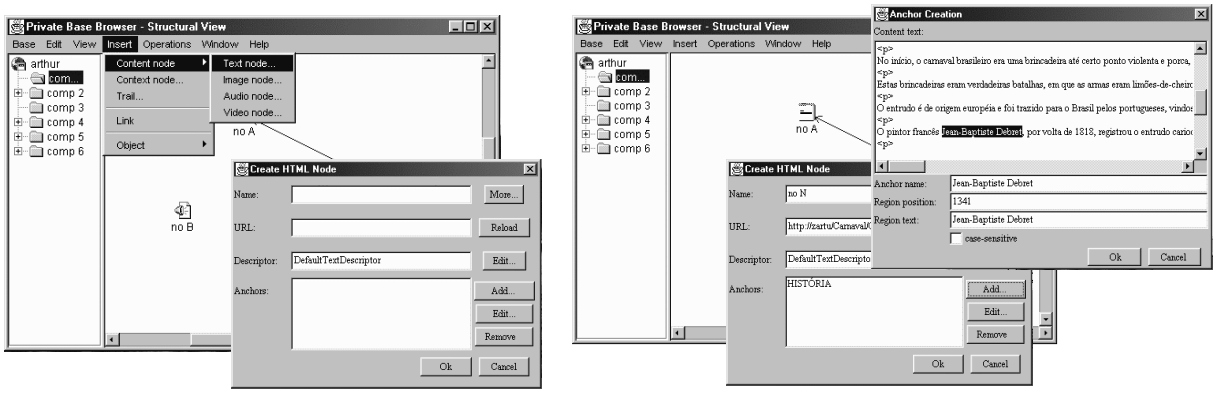
(a) (b)

Figura 6.8 – A visão olho-de-peixe sendo ativada (a) e um nó sendo focalizado (b).

A filtragem olho-de-peixe pode ser reconfigurada pelo ajuste de alguns parâmetros (***configure***): o grau de detalhe desejado e a prioridade nos relacionamentos entre nós (por elos ou por contextos). O desenho do mapa pode ser alterado quando algum desses parâmetros é modificado. Se o usuário quiser que determinados nós sempre fiquem visíveis, como forma de orientar-se na navegação pelo documento, esses nós devem ser marcados como nós de referência através da opção (***landmark***).

A opção restante do menu *visualização* refere-se à visão em forma de árvore da estrutura hierárquica dos documentos. Selecionando-se alternadamente esta opção (***tree view***), a visão em árvore é exibida ou não.

As operações de edição são a principal característica deste navegador e incluem a criação, alteração e remoção de todos os componentes de documentos hipermídia baseados no modelo NCM. O usuário pode criar nós terminais (***insert terminal node***) nas diversas mídias suportadas, através de uma interface bastante fácil: são informados o nome do nó, a localização (URL) do conteúdo, o decritor associado ao nó e suas âncoras. A criação de âncoras é feita marcando-se a região desejada com o *mouse* numa tela que mostra o conteúdo do nó. A Figura 6.9 mostra as fases do processo de criação de um nó terminal texto.



(a) (b)

Figura 6.9 – A criação de um nó terminal texto, informando as propriedades do nó (a) e definindo as âncoras do nó (b).

A criação de nós contexto do usuário (***insert context node***) também segue o mesmo modelo. Além do nome do nó, devem ser informados seus nós componentes. Isto pode ser feito de duas maneiras: selecionando previamente os nós no mapa da base privada, antes de se chamar a caixa de diálogo de criação de contextos, ou através de uma lista de nós contidos na base, que é apresentada para o usuário[17]. Permite-se ainda a remoção de elos deste contexto, mas esta operação só pode ser feita quando o nó for editado, uma vez que a criação de elos é realizada separadamente. A Figura 6.10 apresenta os passos para a criação de um nó de contexto.

[17] Os nós que aparecem nesta lista não podem conter (em qualquer nível de profundidade) o contexto em que serão inseridos. Esta restrição é melhor observada na edição do nó de contexto.

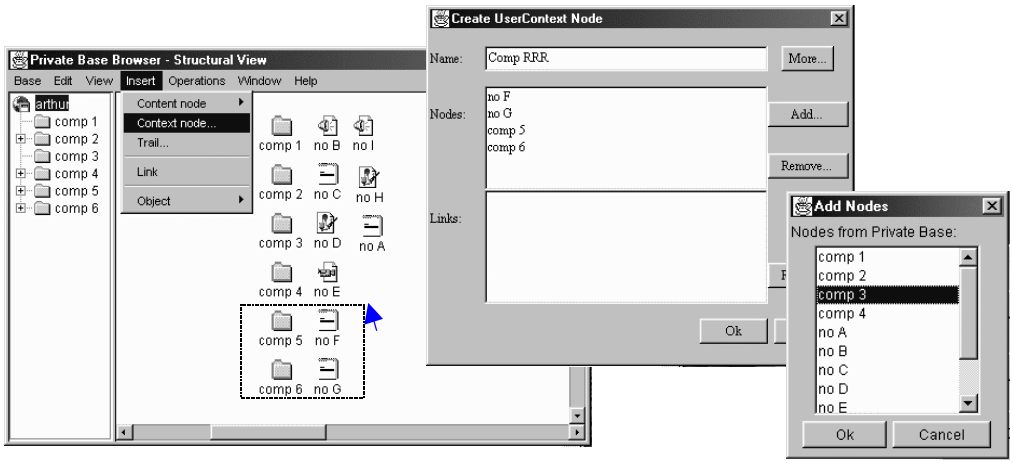
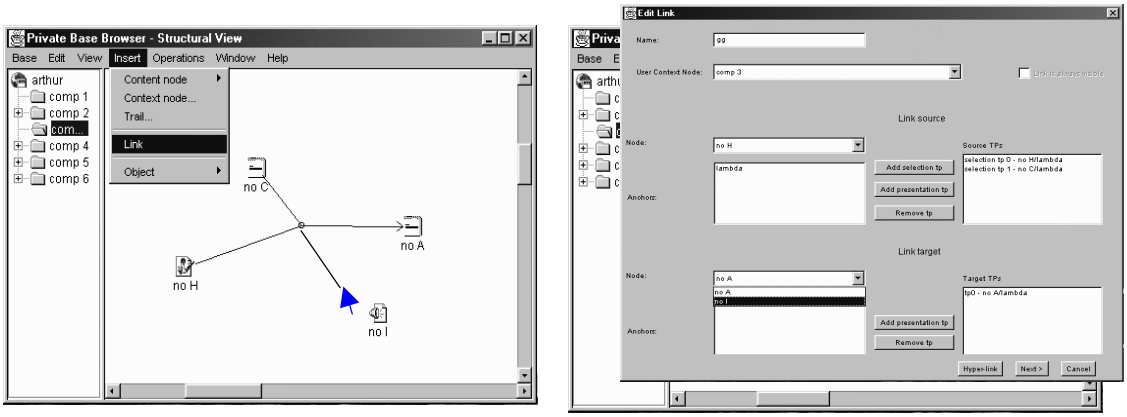


Figura 6.10 – A criação de um nó de contexto. Inicialmente foram selecionados quatros nós para serem componentes do novo contexto. Depois, pela lista de nós da base, o usuário decide incluir mais um nó.

Os elos são criados pela opção de menu (***insert link***). Com o *mouse*, o usuário clica no nó que será a origem do elo e, em seguida, no nó destino do elo. Se for um elo *n*:*m*, a tecla *shift* deve ficar pressionada durante o processo, para que novos ramos possam ser incluídos no elo criado. Esta forma gráfica de criação de elos possibilitaria, por exemplo, a união de elos distintos ou a inclusão de pontos terminais não pertencentes ao contexto do elo, criando elos inconsistentes com o NCM. Um controle rígido é feito na seleção dos nós, para que apenas elos válidos sejam criados ou editados.

Após todos os nós do elo serem escolhidos, a caixa de diálogo de criação de elo é aberta. Deve-se informar o nome do elo, o contexto que irá conter o elo (uma lista com os possíveis contexos é fornecida), os descritores dos nós origem e destino associados ao elo e as âncoras destes nós utilizadas pelo elo. Para completar as informações sobre os pontos terminais do elo, são definidas as condições que devem ser satisfeitas na origem e as ações que serão realizadas no destino. A Figura 6.11 mostra o usuário desenhando o elo (a) e entrando com os dados necessários para a criação do elo (b).



(a) (b)

Figura 6.11 – A criação de elos em dois momentos: desenhando o elo com o auxílio do mouse (a) e determinando as informações necessárias na caixa de diálogo (b).

Como já mencionado, a edição de todas as propriedades (***properties***) dos componentes de um documento utiliza as mesmas interfaces (caixas de diálogos) da criação, simplificando o processo de autoria para o usuário. É importante ressaltar que qualquer modificação em um nó ou elo é propagada automaticamente para todas as suas outras instâncias, mantendo a consistência do documento. Se um elo de uma composição for removido (***delete***), por exemplo, ele será apagado de todas as instâncias desta composição contidas na base privada. A remoção de um nó acarreta na eliminação também dos elos ancorados nele. O sistema permite apagar todo o conteúdo de um elo ou somente um de seus ramos (se for um elo *n*:*m*). Neste caso, o elo pode passar de multiponto para ponto-a-ponto.

Outro elemento que pode ser criado e editado é a trilha. Para se criar um nó trilha (***insert trail***), é necessário informar em uma caixa de diálogo o nome da trilha, seu contexto associado (a composição cujos nós componentes poderão ser incluídos na trilha) e a lista ordenada de nós que forma efetivamente a trilha. A ordem dos nós é determinada pela ordem de seleção dos nós desejados do contexto associado. Pode-se incluir também o conteúdo de outras trilhas de um mesmo contexto associado.

A edição do conteúdo da trilha é feito pela mesma interface de criação ou graficamente, através da manipulação de elos e nós no grafo que representa o conteúdo da trilha. Neste caso, um controle deve existir para assegurar que o grafo formado seja coerente com a definição da trilha. Um algoritmo deve verificar se o grafo contido na trilha não apresenta ciclos e se todos os nós possuem um elo partindo e outro chegando (exceto as extremidades da cadeia). A Figura 6.12 mostra as caixas de diálogo para a criação de trilhas.

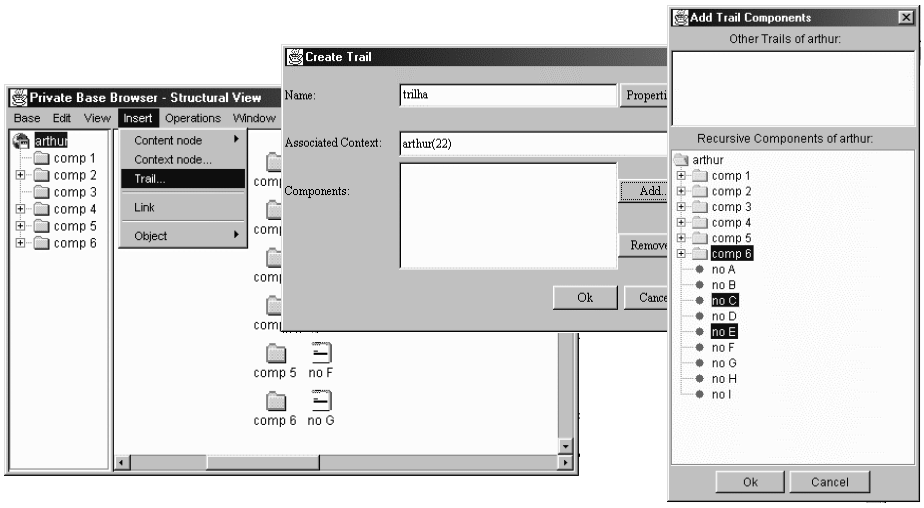


Figura 6.12 – A criação de um nó de trilha. É necessário escolher quais nós do contexto associado serão inseridos na trilha.

No NCM, as trilhas não são armazenadas na hiperbase pública, e sim em um outro repositório próprio no servidor. Assim, trilhas são transferidas do servidor para o cliente (***open trail***) e vice-versa (***save trail***), através de métodos específicos pertencentes ao menu *operações*. No primeiro caso, uma lista com as trilhas existentes no servidor, com seus respectivos nós componentes, é mostrada para o usuário escolher o que deve ser trazido.

O menu *operações* reúne também as funções de ativação de nós e as operações de versionamento. O usuário pode abrir (***expand***) e fechar (***collapse***) as composições selecionadas no mapa, para visualizar ou não seu conteúdo. Pela opção de menu (***play***), o usuário pode iniciar a apresentação de um nó, acionando um exibidor próprio ou o navegador de contextos.

Esta mesma ação é chamada pelo duplo clique do *mouse* sobre um nó no mapa de documentos.

Pela opção (***check in***), o usuário move seus documentos para a hiperbase pública, podendo resgatá-los mais tarde, por meio da operação (***check out***). Quando um nó é transferido para a hiperbase pública, ele é removido do mapa da base privada (se for um nó de composição, seus componentes também são). É interessante que apenas nós estáveis sejam armazenados na hiperbase, pois lá estarão disponíveis para os demais usuários do sistema. Uma vez na hiperbase pública, estes nós não podem ser abertos nem ter seu conteúdo alterado sem que se criem versões. Pelo navegador de base privada, o *check out* dos nós da hiperbase é realizado através de uma caixa de diálogo por onde o usuário entra diretamente com a identificação do nó[18].

Por fim, existe o menu *janelas*, responsável pela ativação dos outros navegadores do sistema. A partir do navegador de base privada, o usuário pode abrir os navegadores de hiperbase pública, de base de contextos de versões, de contexto de versões e o analisador de *web sites*, que cria composições NCM a partir de um endereço WWW.

6.2.2 Navegador de Hiperbase Pública

Como já mencionado, o navegador de hiperbase pública não possui funções de autoria nem de apresentação de documentos. Por ser utilizado basicamente para a localização e transferência de documentos, apresenta apenas as ferramentas de navegação, de visualização da estrutura e de versionamento. Assim, por exemplo, são idênticas às funções do navegador de base privada, as opções de menu relativas à visão olho-de-peixe e ao *layout* automático. Na

[18] Na implementação atual do sistema, é fornecido o nome ou um identificador interno do nó para localizá-lo na hiperbase pública, mas pretende-se futuramente utilizar a URI como forma de identificação.

implementação atual, o navegador mostra todos os nós da hiperbase pública, pois o mecanismo de controle de acessos ainda não foi desenvolvido.

Quando um nó é encontrado, ele deve ser transferido para a base privada para que possa ser manipulado. No navegador de hiperbase, isto é obtido através das opções (***check out***) e (***open***), aplicadas ao nó selecionado no mapa da hiperbase pública. A diferença entre elas está no tratamento de versionamento de composições. Enquanto a primeira traz apenas a composição para a base privada, a segunda cria versões de todos os componentes da composição recursivamente.

Outra função importante oferecida por esse navegador é a de tornar obsoletos os nós armazenados na hiperbase (***obsolete***). No NCM, um nó (OA) contido na hiperbase pública não pode ser destruído diretamente. O usuário deve torná-lo obsoleto primeiro, permitindo que outros nós que o referenciem (por elo ou inclusão), ou dele derivados, sejam alertados. Um nó obsoleto é removido automaticamente por um processo de coleta de lixo do sistema [31], quando sua destruição não provocar inconsistências no hiperdocumento.

A opção (***refresh***) é necessária para periodicamente atualizar o mapa da hiperbase pública com as modificações realizadas por outros usuários do sistema. O navegador conecta-se ao servidor e lê as informações dos nós registrados na hiperbase. Com a implementação do controle de notificação do HyperProp [36], o mapa da hiperbase pública é modificado automaticamente. Porém, esta função permanece no navegador para permitir uma atualização explícita por parte do usuário.

6.2.3 Exibidor de Contextos

O exibidor de contextos é o que apresenta o menor número de funções, uma vez que ele é utilizado apenas para a apresentação de nós de contexto. O exibidor não possui uma barra de menu, e sim um menu *popup*, acionado pelo clique do botão direito do *mouse* sobre o

mapa de documentos. Este menu apresenta apenas as funções de navegação em profundidade (***go down***) e (***go up***), para exibir o conteúdo de um contexto filho ou do contexto pai, respectivamente. A exibição de uma nova composição muda todo o mapa, pois ele mostra apenas um nível da estrutura do documento. Um duplo clique sobre um nó terminal, ativa um exibidor para apresentar o conteúdo deste nó.

Capítulo 7

Considerações Finais

Este último capítulo é dedicado a uma avaliação geral desta dissertação. Em primeiro lugar, são expostos e comparados alguns trabalhos relacionados aos problemas de desenho e filtragem de grafos. Em seguida, os aspectos relativos à interface gráfica de navegação e de autoria da estrutura de documentos dos sistemas analisados no Capítulo 2 são comparados com a solução apresentada nesta dissertação para o sistema HyperProp. Nas conclusões, é feita uma revisão dos objetivos e contribuições desta dissertação, bem como uma crítica dos resultados obtidos. Por fim, são propostas algumas sugestões para trabalhos futuros.

7.1 Comparação com os Trabalhos Relacionados

7.1.1 Desenho de Grafos

Como foi visto, o HyperProp utiliza um mapa ativo, na forma de um grafo composto, como interface gráfica para auxiliar os usuários do sistema durante a navegação no espaço de

informações e na autoria de documentos. Este mapa reúne, em um única visão, tanto a estrutura hierárquica dos documentos como seus relacionamentos hipertexto.

Uma outra abordagem para a visualização de grandes hierarquias é usar a geometria hiperbólica [28]. Os grafos, especialmente árvores, são dispostos num espaço não-Euclideano e mapeados no espaço Euclideano familiar, utilizando um dos modelos da geometria hiperbólica [22]. O resultado produz uma visão distorcida, semelhante ao efeito provocado pelas lentes olho-de-peixe.

Os mapas hiperbólicos sempre exibem a árvore inteira, focalizando uma parte específica, localizada no centro do desenho. À medida que se afastam do foco, os nós são desenhados com dimensões menores e em maior número, como pode ser visto na Figura 7.1 (a). Ao focalizar um outro nó, este é posicionado no centro e o desenho atualizado apropriadamente. O *layout* dos nós de uma árvore no plano hiperbólico é um problema fácil, porque a circunferência e a área de um círculo crescem exponencialmente com seu raio, ou seja, existe mais espaço disponível para alocar os nós. Como uma árvore tende a se expandir também exponencialmente com a profundidade, seus nós podem ser dispostos de uma maneira uniforme, de modo que a distância (medida na geometria hiperbólica) entre os nós permaneça aproximadamente a mesma em qualquer ponto da hierarquia. Na Figura 7.1 (b), a mesma estratégia é estendida para uma visão três dimensões [37].

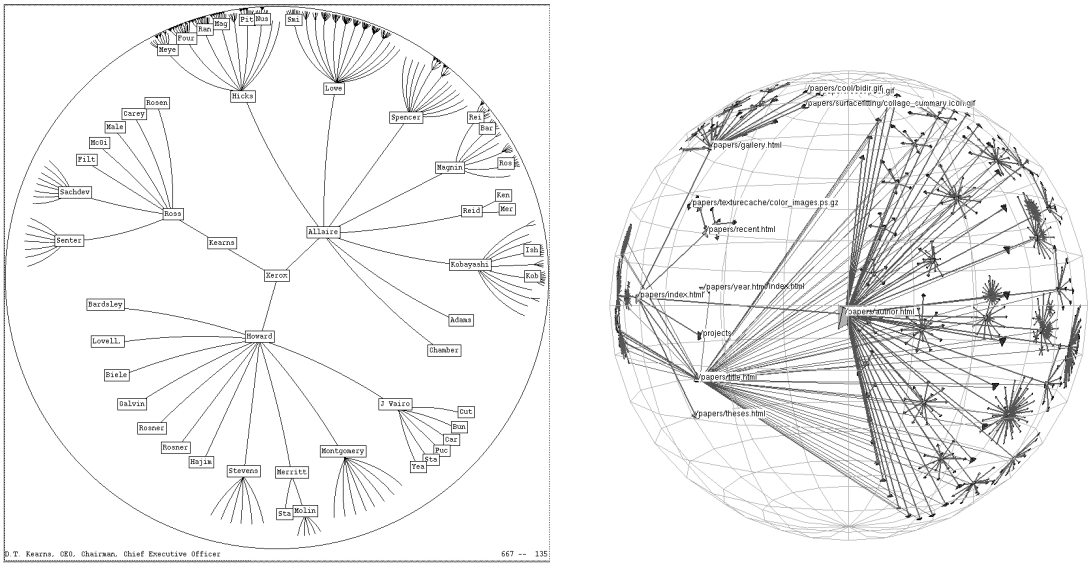


Figura 7.1 – Visão Hiperbólica de um árvore em 2D (a) e 3D (b).

As técnicas que desenham grafos em 3D têm se tornado bastante populares, porque, dentre vários motivos, a dimensão extra pode dar, literalmente, mais “espaço” para exibir grandes estruturas. A árvore em cone (*cone tree*) é uma das melhores técnicas de *layout* de grafos em 3D conhecidas para a visualização de grafos (no caso, árvores) [22]. A árvore em cone foi desenvolvida diretamente para 3D, em vez de ser derivada de um algoritmo 2D, como na maioria das técnicas de desenho de grafos em três dimensões. O *layout* é bastante simples: os nós são posicionados no vértice de um cone, com seus nós filhos uniformemente dispostos ao longo da base. Cones de uma mesma camada têm a mesma altura, com o diâmetro da base sendo reduzido a cada camada. A Figura 7.2 ilustra um exemplo da utilização da árvore em cone na visualização de informações estruturadas [20]. A interatividade e os aspectos visuais são essenciais para tornar as árvores em cone realmente úteis, pois, além dos rótulos dos nós serem transparentes, para permitir que os nós que se localizam atrás sejam visto, o usuário pode girar o cone para trazer para frente o nó escolhido.

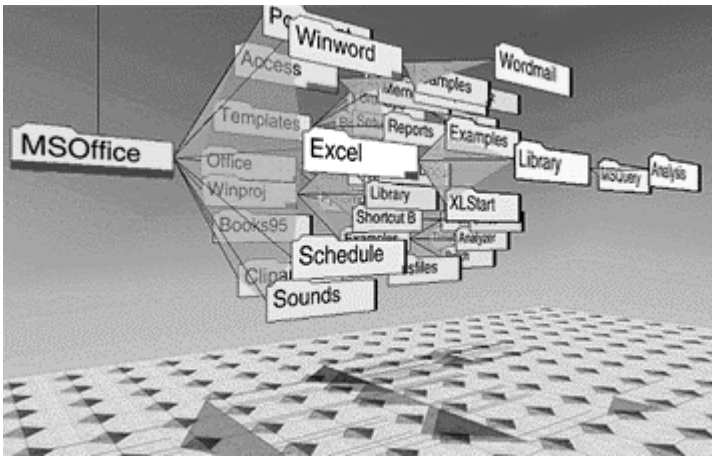


Figura 7.2 – Árvore em cone.

A geometria hiperbólica e a árvore em cone fornecem uma solução elegante para o problema de visualização de hieraquias com um grande número de nós, mas são próprias para estruturas em forma de árvore, não se adequando plenamente ao *layout* de estruturas de documentos hipertexto. Em [22], propõe-se que sejam encontradas as árvores geradoras dos grafos e aplicados tais algoritmos, uma vez que apresentam uma complexidade menor do que os algoritmos de *layout* de grafos. Entretanto, as árvores não conseguem representar, com a clareza do grafo composto, as relações de hieraquia e os elos referenciais dos documentos hipermídia.

7.1.2 Filtragem de Nós

Além de um certo limite no múmero de nós, nenhum algoritmo garante um *layout* adequado para grafos. Simplesmente não há espaço suficiente na tela para alocar tanto nós. Do ponto de vista cognitivo, também não faz sentido exibir uma grande quantidade de informação simultaneamente. Assim, uma etapa preliminar da visualização da estrutura de documentos é a redução do tamanho do grafo a ser exibido. Com isso, algoritmos de *layout* clássicos ainda podem ser aplicados, com resultados satisfatórios.

A visão olho-de-peixe, adotada pelo sistema HyperProp, é um mecanismo de filtragem que se baseia na própria estrutura hierárquica e referencial dos documentos para tentar determinar os nós relevantes para o usuário, de acordo com sua posição no espaço de informações. A *importância a priori* (*API*) dos nós, componente do cálculo da visão olho-de-peixe, se constitui num fator flexível do algoritmo. De fato, escolhendo diferentes funções *API*, obtém-se visões que dão ênfase a padrões estruturais distintos [9]. Por exemplo, o número de vezes que um nó é visitado pode ser utilizado para criar mapas onde os nós mais populares são destacados para um acesso mais fácil.

Além da estratégia olho-de-peixe, existem outras técnicas que reduzem a complexidade dos diagramas estruturais de documentos hipermídia, de tal forma que apenas a informação desejada seja exibida. Em [34], é apresentada uma ferramenta gráfica (*Navigational View Builder*) para a visualização de documentos WWW, que utiliza alguns mecanismos para a filtragem de nós. Os nós são organizados primeiramente em grupos hierárquicos, baseados em dois tipos de análises: da estrutura, onde os elos hipertexto são considerados, ou do conteúdo dos nós, de acordo com similaridades de seus atributos. A filtragem é realizada seguindo critérios semelhantes: por conteúdo do nó, estrutura do documento ou tipo de elo. Por exemplo, podem ser exibidos/ocultados no mapa apenas os nós cujos atributos satisfaçam uma certa propriedade.

A dificuldade deste tipo de filtragem baseada no conteúdo, segundo os próprios autores, é a falta de semântica dos documentos WWW e a definição de atributos úteis para serem incorporados no sistema. Por exemplo, a data do último acesso do arquivo e o número de vezes que um elo foi percorrido são informações que podem ser aproveitadas. A filtragem por atributos, entretanto, é bastante interessante e deve ser cogitada no projeto do HyperProp,

pois, dentre outros motivos, possibilita a introdução de linguagens de consulta no sistema para a busca de documentos.

Uma técnica para a criação de visões *focus+context* de *web sites* é descrita em [35]. A visão *focus+context*[19] fornece detalhes da vizinhança do nó corrente e mostra sua posição em relação a outros nós de referência (*landmarks*) no espaço de informações. Dois tipos de visões são propostos: um diagrama local, que mostra o nó corrente, os nós diretamente ligados a ele e o menor caminho até os nós de referência; e uma visão global, que apresenta somente os nós de referência e os elos entre eles. A existência de mapas estruturais distintos força o usuário a integrá-los mentalmente, o que pode confundir mais do que orientar sua navegação. Entretanto, a forma proposta para se calcular os nós de referência é interessante. Eles são determinados automaticamente por meio de uma análise da freqüência de acesso aos nós e da estrutura do documento (são considerados mais importantes, os nós com alta conexidade e aqueles localizados mais externamente na hierarquia do documento).

7.1.3 Navegadores de Sistemas Hipermídia

No Hyper-G [26], apresentado no Capítulo 2, existem navegadores específicos para cada visão: o *session manager* (Figura 2.2) e o mapa em 3D *information landscape* (Figura 2.4) exibem a estrutura hierárquica das coleções, enquanto que o mapa local (Figura 2.3) mostra os elos hipertexto entre os nós. A cada novo nó selecionado numa visão, a configuração de todos os outros navegadores é atualizada. Mesmo assim, esta divisão é ruim, pois, como já mencionado, a sobrecarga cognitiva para unir as três visões dificulta a compreensão da organização global do documento. Além disso, informações importantes são

---

[19] A visão *focus+context* é uma abordagem comumente empregada quando se quer balancear tanto o detalhe local quanto o contexto global na visualização de uma informação qualquer. Um exemplo desta técnica é a visão olho-de-peixe.

perdidas, como, por exemplo, o contexto dos nós diretamente ligados ao nó selecionado no mapa local.

O Hyper-G procura minimizar este problema de duas maneiras. Uma delas é a reconfiguração do mapa local para poder representar as relações de inclusão entre os nós. Embora esta opção não possa ser usada simultaneamente com o grafo usual de elos referenciais, este recurso mostra-se bastante útil, especialmente quando os nós estão incluídos em mais de uma coleção. A outra maneira é estender o *information landscape* para também mostrar os elos de referência, neste caso, em uma única visão combinada. Os nós são dispostos num plano vertical, de tal modo que aqueles que possuem elos para o documento selecionado ficam posicionados abaixo deste nó, e os nós para os quais este faz referência são mostrados acima [2]. A Figura 7.3 ilustra esta visão. Vê-se que, dependendo do número de nós visíveis, o desenho final pode tornar-se bastante confuso.

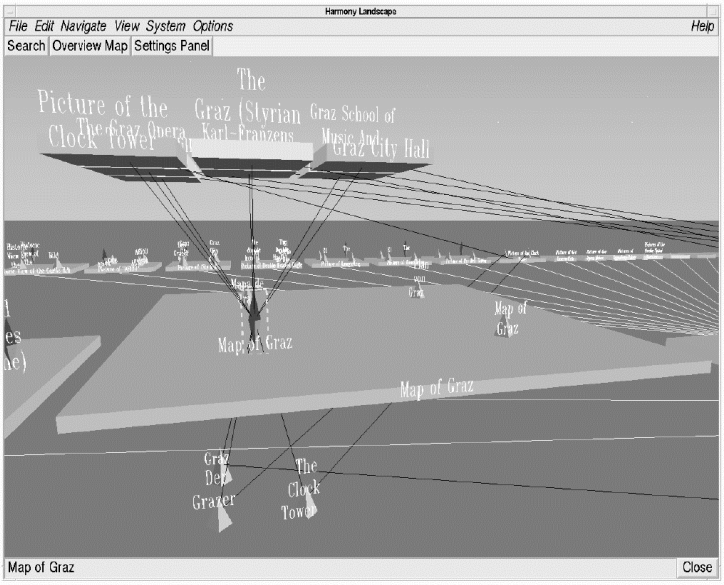


Figura 7.3 – Information landscape exibindo elos referenciais e hierárquicos.

O *layout* dos grafos exibidos pelos navegadores do HyperProp é obtido a partir de um algoritmo do tipo *force-directed*, que revela proximidades entre nós e simetrias no desenho,

baseando-se em critérios estéticos bem definidos. O mapa local do Hyper-G é centrado no nó selecionado pelo usuário, de modo que seus nós adjacentes fiquem organizados em colunas, de acordo com o sentido dos elos (preferivelmente, da esquerda para direita). Esta estratégia pode gerar grafos com elos longos e muito recortados. Além disso, os nós são fixos no mapa, não podendo ser arrastados com o *mouse*, como é permitido nos navegadores do HyperProp.

A filtragem de informação é realizada no Hyper-G através da configuração da quantidade de níveis exibidos no mapa local e pela noção natural de distância no mapa em 3D. A visão olho-de-peixe implementada no HyperProp é mais elaborada, pois leva em consideração a organização hieráquica e relacional do documento para determinar quais nós que serão descartados.

Outra vantagem dos navegadores do HyperProp é permitir a edição gráfica dos componentes de um documento. Os principais mapas gráficos do Hyper-G não apresentam ferramentas de autoria de documentos, servindo apenas para orientar o usuário durante a navegação. A edição é realizada somente no *session manager* (inserção de documentos em coleções) e nos exibidores de documentos (criação de âncoras e elos).

O Kaomi [24], também visto no Capítulo 2, é uma poderosa ferramenta para a construção de ambientes de autoria de documentos multimídia. É composto por diferentes visões dos documentos, todas com seleção e edição sincronizadas, de tal forma que qualquer alteração em uma delas é automaticamente propagada para as outras visões. Sua grande vantagem é a flexibilidade, que permite a criação de novas visões e o suporte a diversos formatos declarativos de documentos.

A principal funcionalidade desta ferramenta é a edição temporal dos documentos. Pode-se iniciar a apresentação de um nó e interrompê-la em qualquer ponto, para se criar um elo, por exemplo. Como a interface gráfica do navegador temporal do HyperProp ainda está

em fase inicial de desenvolvimento, o Kaomi é, sem dúvida, um importante modelo. Pode-se cogitar inclusive a idéia de pesquisar a utilização da própria ferramenta na criação de uma versão alternativa do HyperProp.

A possibilidade de editar o arquivo fonte de um documento e ver as modificações refletidas imediatamente nas outras visões é uma funcionalidade interessante oferecida pelo Kaomi. Uma interface semelhante a esta pode ser pensada para o sistema HyperProp, baseada na linguagem declarativa NCL [4]. Outro recurso que pode ser aproveitado é o suporte às operações de "copiar e colar" nós, tornando assim ao HyperProp um completo ambiente de autoria de documentos, tal qual o Kaomi.

O Kaomi possui como uma de suas visões padrão, a clássica estrutura em árvore para representar a organização lógica dos documentos (Figura 2.5). Por outro lado, a visão relacional não é originalmente oferecida pela ferramenta, tendo que ser criada pelo projetista. Em [24], este é o exemplo de visão dado pelos autores, que pode ser adicionada ao sistema e valer-se da capacidade de edição e sincronismo do Kaomi. Contudo, não há qualquer menção sobre mecanismos de filtragem de nós nem algoritmos de *layout* de grafos.

O *software* comercial Front Page [29] é o mais simples dos sistemas estudados, pois, além de ser baseado no modelo de dados da WWW, é destinado a usuários pouco familiarizados com os padrões da Internet. Entretanto, possui funções poderosas para a criação e administração de *web sites*. A principal vantagem do Front Page é o seu editor HTML, que possibilita uma rápida e fácil criação/edição de páginas WWW (Figura 2.6). Pode-se alterar diretamente o arquivo fonte ou manipular graficamente os elementos contidos na página. Em uma outra visão, que exibe o documento como é apresentado pelos navegadores WWW, pode-se ter uma idéia prévia do trabalho que está sendo feito.

O Front Page oferece vários navegadores, que mostram diferentes visões dos documentos. A organização lógica é baseada na própria estrutura de diretórios de onde as páginas estão armazenadas. Este artifício é ruim, pois não introduz nenhuma semântica ao documento e nem pode ser reutilizado (sem haver duplicação do conteúdo do diretório). No HyperProp, baseado num modelo de maior poder de expressão, o nó de composição tem essa função, podendo conter um número arbitrário de nós e podendo estar contido em várias outras composições. Uma outra visão do Front Page mostra a organização navegacional do documento, apresentando como as páginas de um mesmo documento estão relacionadas entre si. Os elos de navegação são definidos automaticamente pelo usuário, podendo selecionar os nós do mesmo nível de hierarquia, da hieraquia pai ou seus nós filhos.

O mapa de hiperelos, também oferecido pelo Front Page, serve para se ter uma noção de quais páginas estão conectadas ao nó em foco. O *layout* do desenho (Figura 2.8) é similar ao mapa local do Hyper-G: à direita são posicionadas, em uma coluna, as páginas nas quais ancoram os elos que partem do nó em foco, e as páginas cujos elos referem-se a este nó ficam em uma coluna à esquerda. Assim, para cada página, pode-se escolher em mostrar ou ocultar seus nós adjacentes, criando uma estrutura ramificada. Evita-se a sobrecarga do desenho, restringindo a apenas um por coluna o número de nós que podem ser expandidos. A principal falha deste mapa é o desenho de nós repetidos, devido à condição da estrutura hipertexto exibida ser necessariamente uma árvore. Observou-se que se for criado, por exemplo, um elo partindo de uma página *A* para uma página *B* e outro elo no sentido inverso, esses elos são desenhados no mapa indefinidamente.

Uma característica deste mapa é que os elos entre os nós visitados pelo usuário são desenhados de cor diferente para destacar o caminho percorrido. A idéia de modificar a cor dos elementos do mapa é interessante e pode ser usada em diversas situações nos navegadores

de HyperProp, como, por exemplo, para indicar que um nó visível na base privada está localizado na hiperbase pública ou que um determinado nó está sendo editado.

Uma importante ferramenta do Front Page é o analisador de elos, cuja função é manter a integridade do documento, pesquisando o conteúdo de todas as páginas a procura de inconsistências nos elos. Todos os elos do documento são listados e suas páginas de destino, verificadas. Caso uma página não seja encontrada, o elo é marcado como "quebrado" e se, por exemplo, sua URL estiver incorreta, pode-se editá-la uma única vez, que será atualizada em todas as instâncias deste elo.

7.2 Conclusões

O objetivo principal deste trabalho foi definir e implementar uma ferramenta gráfica para a navegação e autoria da estrutura de documentos hipermídia para o sistema HyperProp. Técnicas para desenho e filtragem de grafos foram estudadas para tornar o mapa de documentos mais legível para os usuários do sistema. Através de navegadores gráficos, o usuário tem uma visão global do espaço de informações hipermídia e, utilizando recursos que indicam sua posição em determinado instante, tem condições de saber as possibilidades de navegação. As ferramentas para a edição da estrutura de documentos, desenvolvidas neste trabalho, aproveitam a interface gráfica dos navegadores formada por nós e elos, tornando mais fácil e intuitiva as operações de autoria de estruturas hipermídia.

Como forma de resolver o problema da visualização da estrutura de documentos hipertexto baseados no modelo de composições aninhadas, foi proposta uma extensão do algoritmo de *layout spring-embedder* para grafos compostos. O algoritmo apresentado é aplicado recursivamente em cada nível da estrutura do grafo, que deve ser modificado para

satisfazer certos requisitos: ser não-direcionado e totalmente conexo. Algoritmos diferentes foram desenvolvidos para o *layout* de bases e trilhas, levando-se em conta a inexistência de elos nas primeiras e a ordenação dos componentes das segundas. A grande desvantagem do *spring-embedder* é o seu alto tempo de processamento para grafos com um grande número de nós. A limitação da aplicação do algoritmo para o conteúdo de uma composição por vez e o uso de técnicas de filtragem de nós procuram minimizar essa deficiência.

Como mecanismo de filtragem de nós, foi adotada a visão olho-de-peixe para tentar resolver o clássico problema de desorientação enfrentado pelos usuários de sistemas hipermídia. A técnica, estendida para modelos hipermídia que permitem composição aninhadas de nós [33], foi empregada nos navegadores do sistema de modo que, em uma única visão, exibissem detalhes do nó focado pelo usuário, sem perder a noção da estrutura global dos documentos. A quantidade de informação mostrada pode ser controlada pelo usuário através de ajustes de alguns parâmetros do algoritmo. Assim, a visão olho-de-peixe, aliada ao encapsulamento natural de nós obtido pelo aninhamento de composições, garante a legibilidade dos mapas de documentos.

Por fim, foram definidos todos os navegadores gráficos que compõem os ambientes para autoria e apresentação de estruturas de documentos do HyperProp. Uma vez solucionados os problemas de *layout* e filtragem dos mapas de nós e elos, faltava implementar as ferramentas de edição do sistema e definir a interface entre os ambientes. A edição gráfica dos componentes do documento é totalmente integrada ao mapa estrutural, o que permite um modo mais rápido, fácil e eficaz de autoria hipermídia. A comunicação entre os dois ambientes é um recurso importante para manter a integridade do documento e melhorar a orientação do usuário ao explorar os diferentes navegadores.

Atualmente, um protótipo do sistema HyperProp está sendo utilizado pela equipe de alunos do Laboratório TeleMídia da PUC-Rio e permanece em contante evolução, incorporando novas funcionalidades. Dos testes em andamento, espera-se ganhar experiência sobre o comportamento da interface na prática e as necessidades que possam surgir durante a utilização do sistema.

7.3 Trabalhos Futuros

No âmbito do tema desta dissertação, algumas propostas para trabalhos futuros podem ser sugeridas, como, por exemplo, a melhora no desempenho do algoritmo de *layout* de grafos. O *spring-embedder* precisa inicialmente calcular o menor caminho para todo o grafo, o que aumenta de forma considerável seu esforço computacional se o grafo tiver muitos nós e elos. Técnicas de filtragem de nós e a restrição de ser aplicado em um único nível do grafo por vez procuram melhorar o desempenho do algoritmo de *layout*. Estes artifícios, embora bem sucedidos, nos motivam para futuros estudos com o objetivo de encontrar algoritmos mais rápidos [52].

Recursos gráficos que forneçam alguma informação para o usuário também devem ser explorados. Por exemplo, desenhar com cores diferentes nos mapas, os nós já visitados pelo usuário, ou a trilha de elos percorrida, os nós que estão sendo editados por outros usuários, os elos quebrados, os nós que ainda estão na hiperbase pública, etc. Outras pesquisas podem ser direcionadas para o estudo da visualização do espaço hipermídia através de grafos em três dimensões. Em [3], o *spring-embedder* é estendido para este caso. O sistema Hyper-G é pioneiro na representação de documentos hipermídia em estruturas tridimensionais [2,27].

Na implementação da visão olho-de-peixe no sistema HyperProp, para acelerar o cálculo da função *DOI*, pode-se pensar em manter um cache com todos os valores calculados, de maneira que se um nó for novamente selecionado como foco, a função não precise ser recalculada. Existem trabalhos que estendem a filtragem olho-de-peixe para o caso de haver mais de um nó em foco [19]. Esta abordagem é particularmente interessante para o trabalho cooperativo, onde mais de um usuário pode navegar por um mesmo espaço hipermídia, ou ainda para a edição de documentos, no caso, por exemplo, do autor querer forçar a exibição de dois nós para a criação de um elo entre eles.

Em [35], os autores propõem que a escolha dos *landmarks* possa ser feita de forma automática, utilizando algoritmos que consideram a estrutura do documento (selecionando nós que têm muitos elos) e a freqüência de acesso aos nós. Em [45], a visão olho-de-peixe atua também no tamanho dos nós e na distância entre eles. À medida que se afastam do foco, os nós vão se tornando menores e mais próximos entre si. Tais técnicas podem ser aproveitadas para os mapas estruturais do HyperProp.

Sobre os navegadores desenvolvidos neste trabalho, é fundamental que estes estejam integrados com os outros navegadores do sistema. Como no Kaomi, qualquer atualização no mapa estrutural deve ser propagada nas visões temporal e espacial do documento. No ambiente de autoria, por exemplo, pode-se oferecer o recurso de editar o arquivo fonte do documento, e ver as modificações automaticamente repassadas para o grafo estrutural. Todo o módulo cliente do HyperProp foi projetado para ser extensível, de modo que possa incorporar novas funções naturalmente. É importante também procurar preservar a facilidade de uso das ferramentas de autoria, sem com isso perder suas funcionalidades.

Referências Bibliográficas

[1] Anderson, K.M. "Integrating Open Hypermedia Systems with the World Wide Web". *Proceedings of the Eighth ACM International Hypertext Conference*, Southampton, Inglaterra, pp. 157-166, 1997.

[2] Andrews, K. "Browsing, Building, and Beholding Cyberspace: New Approaches to the Navigation, Construction, and Visualization of Hypermedia on the Internet". *Tese de PhD*, Institute for Information Processing and Computer Supported New Media, Graz University of Technology, Áustria, Setembro de 1996.

[3] Andrews, K. e Kappe, F. "Soaring Through Hyperspace: A Snapshot of Hyper-G and its Harmony Client". *Proceedings of Eurographics Symposium on Multimedia/Hypermedia in Open Distributed Environments*, pp. 181-191, Graz, Áustria, Junho de 1994.

[4] Antonacci. M.J. "NCL: Uma Linguagem Declarativa para Especificação de Documentos Hipermídia com Sincronização Temporal e Espacial". *Tese de Mestrado*, Departamento de Informática, PUC-Rio, Abril de 2000.

[5] Balasubramian, V. "State of the Art Review on Hypermedia Issues and Applications". *Technical Report*, Graduate School of Management, Rutgers University, New Jersey, 1997.

[6] di Battista, G., Eades, P., Tamassia, R. e Tollis, I. "Algorithms for Drawing Graphs: An Annotated Bibliography". *Computational Geometry Theory and*

Applications, 4(5):235-282, 1994.

[7] di Battista, G., Eades, P., Tamassia, R. e Tollis, I. "Graph Drawing". Prentice-Hall, New Jersey, 1999.

[8] Berners-Lee, T.J., Cailliau, R., Groff, J. e Pollermann, B. "World-Wide Web: The Information Universe". *Eletronic Networking: Research, Applications and Policy*, 2(1):52-58, 1992.

[9] Chen, C. "Structuring and Visualising the WWW by Generalized Similarity Analysis". *Proceedings of the 8th ACM International Hypertext Conference*, Southampton, Inglaterra, pp. 177-186, 1997.

[10] Conklin, J. "Hypertext: A Survey and Introduction". *IEEE Computer* 20 (9), pp.17-41, Setembro de 1987.

[11] Cormen, T., Leiserson, C. e Rivest, R. "Introduction to Algorithms". MIT Press, Londres, 1990.

[12] Costa, F.R., Muchaluat, D., Soares, L.F.G. e Souza, G.L. "Editor Gráfico para Estrutura e Sincronismo de Documentos Multimídia". *Anais do IX SIBGRAPI*, pp. 289-296, Caxambú, Brasil, Outubro de 1996.

[13] Cruz, I.F. e Tamassia, R. "How to Visualize a Graph: Specification and Algorithms." Disponível em: http://www.cs.brown.edu/people/rt/gd-tutorial.html.

[14] Davis, H., Hall, W., Heath, I., Hill, G. e Wilkins, R. "Towards an Integrated Information Environment with Open Hypermedia Systems". *ECHT'92, Proceedings of the Fourth ACM Conference on Hypertext*. Milão, Itália, Novembro de 1992.

[15] Davis, H., Heath, I., Cesnik, B. e Vanzyl, A. "Open Hypertext Systems: An Examination of Requirements, and Analysis of Implementation Strategies, Comparing Microcosm, HyperTED and the World Wide Web", *Technical Report*, 1994.

[16] Eades, P. "A Heuristic for Graph Drawing". *Congressus Numerantium* 42,

pp.149-160, 1994.

[17] Eades, P., Feng, Q. e Lin, X. "Multilevel Visualization of Clustered Graphs". *Graph Drawing' 96, Lecture Notes in Computer Science*. Spring-Verlag, 1996.

[18] Furnas, G. "Generalized Fisheye Views". *Proceedings of ACM SIGCHI'86 Conference on Human Factors in Computing Systems*, pp 16-23. Boston, Abril de 1986.

[19] Gutwin, C. e Greenberg, S. "Interactive Fisheye Views for Groupware". *Technical Report*, Department of Computer Science, University of Calgary, 1997.

[20] Hemmje, M., Kunkel, C. e Willet, A. "LyberWorld: A Visualization User Interface Support Fulltext Retrieval". *Proceedings of ACM SIGIR'94*, ACM Press, 1994.

[21] Harel, D. "On Visual Formalisms". *Communications of the ACM*, 31(5):514-530, 1988.

[22] Herman, I., Melançon, G. e Marshall M.S. "Graph Visualisation and Navigational in Information Visualisation". *Eurographics Conference*, Milão, Itália, 1999.

[23] Johnson, D.S. e Pollack, H.O. "Hypergraph Planarity and the Complexity of Drawing Venn Diagrams". *J. Graph Theory* vol. 10(3): 309-325, 1987.

[24] Jourdan, M., Roisin, C. e Tardif, L. "A Scalable Toolkit for Designing Multimedia Authoring Environments". *Multimedia Authoring and Presentation: Strategies, Tools and Experiences Multimedia Tools and Applications Journal, Special Number*, Kluver Academic Publishers, 1999.

[25] Kamada, T. e Kawai, S. "An Algorithm for Drawing General Undirected Graphs". *Information Processing Letters*, 31: 7-15, Abril de 1989.

[26] Kappe, F., Andrews, K., Faschingbauer, J., Gaisbauer, M., Pichler, M. e Schipflinger, J. "Hyper-G: A New Tool for Distributed Hypermedia". *Technical Report* 388, IICM Graz University of Technology, 1994.

[27] Kumar, A. e Fowler, R.H. "A Spring Modeling Algorithm to Position Nodes of an

Undirected Graph in Three Dimensions". *Technical Report*, Departament of Computer Science, University of Texas – Pan American, Edinburg, 1994.

[28] Lamping, J., Rao, R. e Pirolli, P. "A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies". *Human Factors in Computing Systems, CHI'95 Conference Proceedings*, ACM Press, pp 401-408, Denver, EUA, 1995.

[29] Microsoft Press. "Microsoft Front Page 98 – Step by Step". Active Education, 1998.

[30] Misue, K. e Sugiyama, K. "An Overview of Diagram Based Idea Organizer: D-ABDUCTOR". *Technical Report* IIAS-RR-93-3E, ISIS, Fujitsu Laboratories, 1993.

[31] Moreira, M.S. "Servidor do Sistema HyperProp". *Tese de Mestrado, Departamento de Informática, PUC-Rio*, Brasil, a ser defendida em Agosto de 2000.

[32] Muchaluat, D. "Browsers e Trilhas para Documentos Hipermídia Baseados em Modelos com Composições Aninhadas". *Tese de Mestrado*, Departamento de Informática, PUC-Rio, Março de 1996.

[33] Muchaluat, D.C. e Soares, L.F.G. "Fisheye View for Compound Graphs". *Relatório Técnico do Laboratório TeleMídia*, PUC-Rio, Maio de 1998.

[34] Mukherjea, S. e Foley, J. "Visualizing the World-Wide Web with the Navigational View Builder". *Computer Networks and ISDN Systems – Special Issue on the 3rd International World-Wide Web Conference*, 27(6):1075-1087, Darmstadt, Alemanha, 1995.

[35] Mukherjea, S. e Hara, Y. "Focus+Context View of World-Wide Web Nodes". *Proceedings of the 8th ACM International Hypertext Conference*, Southampton, Inglaterra, 1997.

[36] Muniz, B. C. "Notificação e Controle de Versões para o Suporte à Autoria

Cooperativa no Sistema HyperProp". *Tese de Mestrado, Departamento de Informática, PUC-Rio,* Brasil, a ser defendida em Agosto de 2000.

[37] Munzner, T. "Drawing Large Graphs with H3Viewer and Site Manager". *Proceedings of the Symposium on Graph Drawing GD'98*, Springer-Verlag, pp. 384-393, Montreal, Canadá, 1998.

[38] Nelson, Ted. "A File Structure for the Complex, the Changing and the Indeterminate". *ACM 20th National Conference,* 1965.

[39] Noik, E.G. "Exploring Large HyperDocuments: Fisheye View of Nested Networks" *Proceedings of the 5th ACM Conference on Hypertext*, Seatle, EUA, pp. 102-205, 1993.

[40] Open Hypermedia Systems Working Group. "OHSWG Compendium". Novembro de 1997.

[41] Rada, Roy. "Hypertext: From Text to Expertext". McGraw-Hill Publishers, 1991.

[42] Rodrigues, L.M., Rodrigues, R.F., Muchaluat-Saade, D.C. e Soares, L.F.G. "Improving SMIL Documents with NCM Facilities", *Proceedings of the Multimedia Modeling Conference MMM'99,* Canadá, Outubro de 1999. Também publicado em: *V Simpósio Brasileiro de Sistemas Multimídia e Hipermídia*, Goiânia, Junho de 1999.

[43] Rodrigues, R.F., Muchaluat-Saade, D.C. e Soares, L.F.G. "Composite Nodes, Contextual Links and Graphical Structural Views on the WWW". *Special Issue on World-Wide Web of the Brazilian Computer Society* 5(2), Novembro de 1998.

[44] Rodrigues, R.F., Soares, L.F.G. e Souza, G.L. "O Ambiente de Execução do Sistema HyperProp para Apresentação de Documentos Multimídia/Hipermídia". *III Workshop em Sistemas Multimídia e Hipermídia*, São Carlos, Brasil, Maio de 1997.

[45] Sarkar, M. e Brown, M.H. "Graphical Fisheye Views". *Communications of the ACM*, Vol. 37 No 12, Dezembro de 1994.

[46] Sim, S. "Automated Graph Drawing Algorithms". *Algorithms in Graph Theory*, CSC2410F, Fall, 1996.

[47] Soares, L.F.G. "Modelo de Contextos Aninhados – Versão 2.3". *Relatório Técnico do Laboratório TeleMídia*, PUC-Rio, Fevereiro de 2000.

[48] Soares, L.F., Casanova, M.A. e Rodriguez, N.L. "Nested Composite Nodes and Version Control in an Open Hypermedia Systems". *International Journal on Information Systems, Special Issue on Multimedia Information Systems*, 20(6):501-519, 1995.

[49] Soares, L.F.G., Souza, G.L., Rodrigues, R. e Muchaluat, D. "Versioning Suport in the HyperProp System". *Multimedia Tools & Applications*, Vol 8 (8), Maio de 1999.

[50] Sugiyama, K. e Misue, K. "Visualization of Structural Information: Automatic Drawing of Compound Digraphs". *IEEE Transactions on Systems, Man and Cybernetics*, 21(4):876-892, 1991.

[51] Szwarcfiter, J.L. "Grafos e Algoritmos Computacionais" Ed. Campus, Rio de Janeiro, 1984.

[52] Tunkelang, D. "A Numerical Optimization Approach to General Graph Drawing". *Tese de PhD*, School of Computer Science, Carnegie Mellon University, 1999.