Editores Gráficos para Estruturas com Composições[1]

Débora Christina Muchaluat[1]
Luiz Fernando Gomes Soares[1]
Marco Antonio Casanova[2]

[1]Laboratório TeleMídia
Depto. de Informática, PUC-Rio
R. Marquês de São Vicente, 225
22453-900 Rio de Janeiro, Brasil
{debora,lfgs}@inf.puc-rio.br

[2]Centro Científico Rio, IBM Brasil
Caixa Postal 4524
20001-930 Rio de Janeiro, Brasil
casanova@vnet.ibm.com

Abstract

This paper discusses compound graphs graphical editors. Besides offering relations modeled by arcs, compound graphs also offer composition relations modeled by the content of nested subgraphs. When dealing with large and complex structures, it is often impossible to show all the graphs' components simultaneously, since the diagram obtained would be very confusing and illegible. Our proposal presents an algorithm for filtering information using an extension of the fisheye-view strategy for compound graphs. This technique can be applied to many fields where the using of graphical editors is crucial. We present an implementation of our proposal for hypermedia systems, building structural browsers that use and validate the algorithm.

Palavras-chave: grafos, editores, composições, filtros, olho-de-peixe

1 Introdução

Editores gráficos para a manipulação de estruturas de grafos são amplamente utilizados em diversas aplicações, pois facilitam a organização de informações. Uma das dificuldades dos projetistas desse tipo de ferramenta é o fato das estruturas terem tamanho ilimitado. Por isso, grafos grandes e complexos, com vários níveis de aninhamento (para estruturas que permitem composições), exigem editores bem elaborados para a exibição de sua estrutura. Parâmetros como grande número de vértices, grande número de arcos, muitas mudanças no grafo, tempo de

[1] O trabalho descrito neste artigo foi parcialmente financiado pelo projeto HyperProp-PROTEM-II do CNPq

resposta ruim para as ações do usuário, diferenças visuais insuficientes entre vértices e arcos e usuários desorientados visualmente se combinam para dificultar a utilização dos editores gráficos.

A função mais importante dos editores gráficos é fornecer uma interface gráfica para manipulação de estruturas. Em diversas aplicações, o uso desses editores é crucial, pois facilita bastante o trabalho do usuário. Por exemplo, para a criação de um estrutura em grafo, seja na definição das classes de um sistema, dos módulos de uma configuração ou da estrutura de um documento hipermídia, é muito mais fácil fazê-lo através de uma interface gráfica. Não só no momento de autoriana engenharia de software ou engenharia de documentos, mas para a visualização e navegação na estrutura em grafo, as ferramentas gráficas têm sua importância destacada. No caso de sistemas hipermídia, por exemplo, o browser de estrutura é fundamental para facilitar a navegação nos hiperdocumentos e ajudar a reestabelecer o senso de localização do usuário na estrutura, quando ele está desorientado. No caso de ferramentas case, o editor gráfico para o diagrama de classes ou para o diagrama de módulos é bastante eficaz para exibir relacionamentos entre as entidades de um sistema de software. Em diferentes aplicações, encontramos uma funcionalidade para editores gráficos para estruturas em grafo, mas o problema é único: como apresentar a estrutura para o usuário ?

A maneira mais direta de dar aos usuários uma visão estrutural é construir um mapa de todo o grafo, destacando o vértice selecionado pelo usuário e mostrando todos os arcos que chegam ou saem dele. Entretanto, quando a estrutura do grafo em questão é muito complexa, é essencial controlar o número de vértices e arcos exibidos na ferramenta, o que pode ser feito com a utilização de filtros para esconder informações. Neste artigo, apresentamos uma maneira de construir tais editores usando a estratégia de olho-de-peixe proposta inicialmente por Furnas [Furn86], e discutida por vários autores [ToDi92, Gloo91, Noik93, SaBr92]. Propomos uma extensão para essa estratégia para estruturas em grafo que permitem subgrafos aninhados.

Este artigo está organizado da seguinte forma. A Seção 2 revê os principais conceitos de estruturas com subgrafos aninhados, apresenta rapidamente o conceito de olho-de-peixe bem conhecido de Furnas e exibe uma explicação detalhada da nossa extensão para estruturas com subgrafos aninhados. A Seção 3 discute algumas aplicações da técnica de filtragem para a construção de editores, ilustrando com uma implementação desses conceitos para sistemas hipermídia. A Seção 4 resume alguns trabalhos relacionados e compara-os com a nossa proposta. Finalmente, a Seção 5 contém as conclusões.

2 Editores para Estruturas com Subgrafos Aninhados

A definição de grafos é baseada em dois conceitos familiares, chamados vértices e arcos. Existem alguns modelos conceituais que permitem agrupar vértices em vértices de composição e também permitem composições aninhadas. Tais modelos ajudam a organizar a estrutura através de subgrafos aninhados.

Um vértice de composição agrupa vértices e arcos, chamados componentes, incluindo outros vértices de composição, em modelos que permitem aninhamento. Editores gráficos para essas estruturas devem exibir um diagrama de seus componentes, permitindo operações de edição.

A motivação fundamental da estratégia de olho-de-peixe [Furn86] é balancear detalhes locais e o contexto global em relação a um vértice selecionado pelo usuário. Detalhes locais são necessários para dar aos usuários informações relacionadas à sua localização na estrutura do grafo. Contexto global é importante para dar aos usuários informações sobre sua posição dentro da estrutura global em questão.

A estratégia básica do olho-de-peixe propõe uma função de grau de interesse que atribui a cada vértice um valor representando seu interesse para o usuário. Esta função é decomposta em duas componentes. A primeira é chamada de importância a priori e descreve a importância intrínseca de um vértice considerando a estrutura do grafo. A segunda componente determina a distância entre o vértice para o qual a função está sendo calculada e o vértice em foco[2] pelo usuário. A essência da visão olho-de-peixe é que a função de grau de interesse aumenta com a importância a priori (API) e diminui com a distância (D).

Para descrever a visão olho-de-peixe estendida que estamos propondo, precisamos introduzir algumas definições que resultam de modelos com subgrafos aninhados. Já que nestes modelos é permitido que diferentes subgrafos contenham o mesmo vértice e que subgrafos estejam aninhados em qualquer profundidade, precisamos de uma maneira para identificar por qual seqüência de vértices de composição um dado vértice está sendo observado. Isto é dado pela noção de perspectiva de um vértice [SoCR95].

Uma perspectiva de um vértice N é uma seqüência $P = (N_m,...,N_1)$, com $m \geq 1$, tal que $N_1=N$, N_m é um vértice não contido em nenhum outro vértice de composição, N_{i+1} é um vértice de composição e N_i está contido em N_{i+1}, para $i \in [1,m)$, com $m>0$. Já que N é dado implicitamente por P, nós iremos nos referir a P simplesmente como uma perspectiva. Note que podem existir diversas perspectivas para o mesmo vértice N, se esse vértice está contido em mais de um vértice de composição.

Dada uma perspectiva $P = (N_m,...,N_1)$, com $m \geq 1$, nós chamamos de prefixos de P, todas as perspectivas $P_i = (N_m,...,N_i)$, para $i \in (1,m]$.

O grau de interesse do usuário para cada vértice depende da perspectiva pela qual está sendo analisado. O mesmo vértice terá um valor da função de grau de interesse para cada uma de suas ocorrências na estrutura (perspectiva). A função de grau de interesse deve ser calculada para cada perspectiva da estrutura do grafo.

Antes de definir como a função de grau de interesse será calculada, devemos observar a definição de arcos, que serão usados para calcular a distância abstrata entre perspectivas.

Um arco contém um conjunto de pontos terminais origem e um conjunto de pontos terminais destino. Múltiplos pontos terminais origem e destino permitem a definição de relações n-para-m (muitos modelos conceituais, entretanto, só permitem relações 1-para-1).

[2] Dizemos que um vértice está em foco quando é a base para o cálculo da função de grau de interesse para todos os outros vértices.

Cada um dos pontos terminais de um arco é definido por $<(N_k,...,N_2,N_1)>$ onde $N_k,...,N_2,N_1$ é uma lista de vértices. N_{i+1} deve ser um vértice de composição e N_i deve estar contido em N_{i+1}, para todo $i \in [1,k)$ com $k>0$. Também é necessário que, para cada arco l contido em um vértice de composição C, para cada ponto terminal $<(N_k,...,N_2,N_1)>$ de l, o vértice N_k deve ser C ou um vértice contido em C.

Se um arco possui um único ponto terminal origem $<(S_m,...,S_1)>$ e um único ponto terminal destino $<(T_n,...,T_1)>$, é denotado pelo par $(<(S_m,...,S_1)>,<(T_n,...,T_1)>)$ no que se segue. Agora podemos prosseguir com nossa extensão para estratégia de visões olho-de-peixe, focalizando somente arcos 1:1, por simplicidade.

Para descrever o nosso algoritmo de filtragem, é importante definir alguns requisitos que devem ser sempre satisfeitos. Uma discussão melhor destes requisitos pode ser encontrada em [Much96]. Os requisitos são:

1. se o último vértice N_1 em uma perspectiva $P = (N_m,...,N_1)$ é exibido no mapa, todos os vértices de P também devem ser exibidos, para que a noção de composição não seja perdida;
2. dado um arco da forma $(<(N_k,N_{k-1},...,N_1,x)>,<(K_n,K_{n-1},...,K_1,y)>)$, onde y é o vértice em foco em uma perspectiva que contém o arco[3], o vértice N_k é sempre exibido e os vértices $N_k,N_{k-1},...,N_1,x$ têm prioridade decrescente de exibição;
3. se um vértice de composição é vértice em foco, seus componentes são sempre exibidos no mapa.

Para satisfazer a todos os requisitos acima, temos que estender a estratégia olho-de-peixe, definindo os componentes da função de grau de interesse como a seguir:

- a importância a priori de um vértice x em uma perspectiva $P = (N_m,...,N_1,x)$, denotada por $API(P)$ é dada por (-m). Quanto maior o nível de aninhamento de um vértice, menor sua importância a priori;

- a distância entre um vértice x em uma perspectiva $P = (N_m,...,N_1,x)$ para o vértice em foco y em uma perspectiva $Q = (K_n,...,K_1,y)$, denotada por $D(P,Q)$ é calculada pela seguinte função:

$$D(P,Q) = min(D_c(x,y),D_l(x,y))$$

onde $D_c(x,y)$ é a distância mínima entre x e y considerando somente a hierarquia de composições do grafo e $D_l(x,y)$ é a distância mínima entre x e y considerando somente arcos. Se não existirem arcos, definidos em P ou Q conectando x e y, a distância $D(P,Q)$ é igual à distância considerando somente o aninhamento de subgrafos $D_c(x,y)$.

Para calcular a distância considerando vértices de composição $D_c(x,y)$, nós consideramos o aninhamento de subgrafos, contando o número de subgrafos (vértices de composição) de P e Q acessados, para cima e para baixo.

[3] Dizemos que uma perspectiva contém um arco se este arco está definido em um dos vértices de composição da perspectiva.

Para calcular a distância considerando arcos $D_l(x,y)$, consideramos os caminhos entre y e x. Se não existirem arcos diretos de y a x, devemos computar a menor distância considerando todos os caminhos possíveis (usando vários arcos), entre x e y. Note que os arcos são considerados bidirecionais, mesmo para grafos dirigidos.

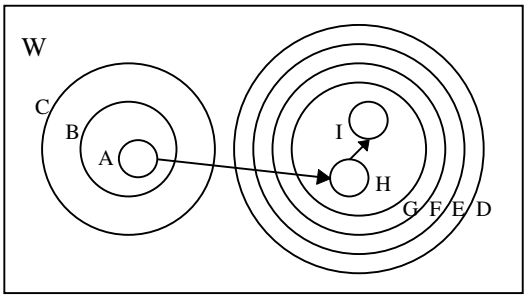
Para um arco com o formato genérico $(<(N_k,...,N_1)>,<(M_n,...,M_1)>)$ com o vértice N_1 em uma perspectiva $P = (C_m,C_{m-1},...,N_k,...N_1)$ e os vértices em $M_n,...,M_1$ nas perspectivas $Q_n = (C_m,C_{m-1},..., M_n)$, $Q_{n-1} = (C_m,C_{m-1},..., M_n,M_{n-1})$, ..., $Q_1 = (C_m,C_{m-1},..., M_n,M_{n-1},...,M_1)$, respectivamente, a distância entre os vértices M_i e N_1, para todo $i \in [1,n]$ com $n>0$ é dada pela expressão:

$$distância = (k - 1) + (n - i + 1)$$

Finalmente, para computar a função de grau de interesse para o vértice x em uma perspectiva $P = (N_m,...,N_1,x)$ no grafo, tendo o vértice y em uma perspectiva $Q = (K_n,...,K_1,y)$ como foco, usamos a seguinte expressão:

$$DOI(P,Q) = API(P) - D(P,Q)$$

Por exemplo, considere um grafo hipotético W com a seguinte estrutura da Figura 1. Para calcular a função de grau de interesse para o vértice A na perspectiva $P = (W,C,B,A)$, tendo o vértice I na perspectiva $Q = (W,D,E,F,G,I)$ como foco, temos:



API(P) = -3
D(P,Q) = min (D_c(A,I),D_l(A,I))
D_c(A,I) = 8 (caminho: B-C-W-D-E-F-G-I)
D_l(A,I) = 8 (caminho: B-C-D-E-F-G-H-I)
então, D(P,Q) = min (8,8) = 8

como DOI(P,Q) = API(P) - D(P,Q)
DOI(P,Q) = -3 -8 = -11

Figura 1 - Exemplo da função de grau de interesse

Depois de calcular as componentes da função de grau de interesse, calculamos o valor da função para cada perspectiva P dependendo da perspectiva em foco Q. A filtragem de informações é feita definindo um valor K que será o limiar para a função de grau de interesse. Somente as perspectivas que têm o valor de grau de interesse maior ou igual a K serão exibidas no diagrama.

Se calcularmos a função de grau de interesse para todas as perspectivas, notaremos que o maior valor da função é dado aos prefixos da perspectiva corrente em foco. Este fato satisfaz nosso primeiro requisito para o mecanismo de filtragem. Para satisfazer os outros requisitos, devemos limitar o valor máximo de K, que define as perspectivas visíveis no mapa. Se considerarmos um exemplo genérico de composição, é simples provar que o valor de K não deve exceder o valor máximo da função de grau de interesse menos 2 unidades, quando o último vértice da perspectiva for um vértice de composição, e nao deve exceder o valor máximo da função de grau de interesse menos 1 unidade, caso contrário, para satisfazer todos os requisitos mencionados anteriormente.

O valor máximo da função de grau de interesse é igual a importância a priori da perspectiva em foco. Para uma explicação detalhada sobre essas condições, deve-se consultar [Much96].

Devemos notar que não é conveniente computar o valor da função de grau de interesse para todas as perspectivas em todos os níveis de aninhamento das composições durante todo o tempo. Somente as perspectivas já visitadas pelo usuário e aquelas no primeiro nível de aninhamento são consideradas para o cálculo. Isso limita o escopo de perspectivas e melhora o tempo de resposta do algoritmo. Além disso, a legibilidade do mapa é melhorada, pois o usuário só visualizará mais detalhes sobre os vértices que realmente lhe interessam. O caso em que todos os vértices são considerados para o cálculo da função de grau de interesse é específico. Isso acontece quando o usuário acessa todos os vértices de composição no browser.

Caso o usuário não se interesse mais em ver os componentes de um vértice de composição no diagrama, ele pode simplesmente pedir uma operação de "implodir" o vértice de composição, que desconsiderará seus componentes para o cálculo da função de grau de interesse.

Já que um mesmo vértice pode ser inserido em diferentes contextos, ele também pode ser desenhado em diferentes posições no mapa. Por outro lado, exibir um vértice em uma de suas perspectivas não significa que será exibido em todas as outras.

Como a posição de um vértice em uma dada perspectiva depende de quais perspectivas são exibidas no mapa, os vértices de um hiperdocumento não têm uma posição fixa na tela. Para reduzir esse problema, o usuário pode definir algumas perspectivas como pontos de referência, i.e., perspectivas que serão sempre exibidas no mapa, pelo menos parcialmente, ajudando o usuário a melhorar o seu senso de localização. Esses pontos de referência são chamados *landmarks* e são específicos a cada usuário, logo um usuário não precisa concordar com o conjunto de landmarks de qualquer outro usuário.

Landmarks serão exibidos dependendo do nível de aninhamento sendo apresentado. Por exemplo, um vértice *N* que é componente de uma perspectiva landmark será exibido se qualquer outro vértice com a mesma importância a priori de *N*, ou maior, for exibido no mapa. Então, para cada perspectiva landmark, pelo menos um de seus vértices é exibido no mapa (aquele com a maior importância a priori).

3 Aplicações

Editores gráficos para estruturas com composições são extremamente úteis em diversas áreas, por exemplo, no projeto de sistemas de software, na navegação em sistema hipermídia, no suporte a linguagens de configuração, entre outros.

No ambiente para modelagem de sistemas de software orientado a objetos fornecido pelo sistema 2GOOD [Carv96], Second Generation Object Oriented Development, é utilizada uma estrutura em grafos aninhados para a definição do diagrama de classes. O editor gráfico para esses diagramas permite que o usuário agrupe conjuntos de classes em composições e transforme-os em classes independentes, modularizando o sistema em vários subsistemas aninhados. Nessa aplicação, as técnicas descritas neste artigo se aplicam diretamente. A princípio, um sistema de software pode ser muito grande, portanto, seu diagrama de classes pode ser representado

graficamente por um grafo complexo, o que implica na utilização de mecanismos de filtragem para a exibição das informações. O algoritmo baseado nas lentes olho-de-peixe seria uma boa proposta pois ainda permite que usuário molde a quantidade de informações exibidas de acordo com seus interesses.

Neste sistema, as composições, ou melhor, os subgrafos, agrupam classes intimamente relacionadas que, após identificar-se a sua funcionalidade, acabam se transformando em uma única classe que pode ser utilizada em qualquer outro sistema.

Outra aplicação de editores gráficos para estruturas com composições aninhadas é em ambientes para programação distribuída utilizando linguagens de configuração [PPBC96].

Um programa distribuído pode ser considerado como um conjunto de processos (vértices) distintos, separados espacialmente, que interagem entre si através de troca de mensagens [Dula90]. O conceito de modularidade, ou seja de se programar sistemas dividindo-os em módulos (composições), já vem de vários anos em Engenharia de Software. Em [ReKr75], encontramos a idéia de Programming-in-the-small X Programming-in-the-large, que consiste basicamente de se separar a programação da funcionalidade do sistema (Programming-in-the-small) de sua estrutura lógica total (Programming-in-the-large). O paradigma de configuração utiliza justamente esta idéia, usando a estrutura lógica e a física na programação da estrutura de um sistema. Assim, o paradigma de configuração procura controlar os módulos de um sistema, suas ligações com os demais módulos e, ainda, a forma como eles estão distribuídos na rede onde se pretende que o sistema seja executado. Um editor gráfico para esse tipo de ambiente exibe a estrutura de uma aplicação, permitindo que o usuário manipule módulos, portas e ligações entre eles.

Outra aplicação de editores para grafos aninhados é a construção de browsers para sistemas hipermídia baseados em modelos com composições aninhadas. Para validar as idéias propostas, implementamos as técnicas de filtragem de informações para a construção de browsers para o sistema hipermídia HyperProp [SoCR95].

O browser é um componente muito importante dos sistemas hipermídia, pois reduz o chamado problema “perdidos no hiperespaço” [Hala88]. A função do browser é mostrar a estrutura de um hiperdocumento ou parte dela como um grafo, dando uma importante medida de contexto e espaço. Isto ajuda ao usuário a saber quais vértices ele está acessando como estão relacionados com seus vizinhos no grafo.

Usar o browser pode ser comparado a quando nós estamos em uma livraria e encontramos um livro que parece interessante. Antes de comprá-lo, damos uma rápida olhada em sua estrutura como um todo e às vezes analisamos alguns capítulos com mais detalhes. Talvez esse seja o livro que estamos procurando, ou talvez ele tenha uma referência para outro livro que pode estar mais relacionado aos nossos interesses. A livraria é o nosso “hiperespaço” e cada livro é um hiperdocumento. O browser oferece uma visão estrutural do “hiperespaço” e dos hiperdocumentos e mostra outros livros relacionados ao que estamos analisando.

Os browsers do HyperProp utilizam o mecanismo para filtrar informações baseado nas técnicas descritas na Seção 2. As ferramentas permitem visualização e edição da estrutura de hiperdocumentos, oferecendo também facilidades para a navegação.

Como o objetivo do HyperProp é construir ferramentas flexíveis, os browsers permitem que o usuário escolha o nível de detalhe que será apresentado no mapa, que define o número de nós e elos exibidos. Então, tanto a rede de interconexões inteira (máximo nível de detalhe) ou somente o que for necessário para dar aos usuários um senso de localização pode ser visualizado.

Para apresentar visões olho-de-peixe de nós de composição, utilizamos uma técnica para layout automático de grafos [SuMi91]. Como a estrutura de nós e elos visíveis muda, o diagrama pode se modificar drasticamente de uma visão para outra. Mudanças rápidas e drásticas nos diagramas normalmente destroem o mapa mental do usuário, diminuindo a eficiência da ferramenta e causando a desorientação. Então, para reduzir esse problema, os browsers mostram mudanças nos diagramas através de técnicas de animação. Esse recurso reduz a mudança visual instantânea preservando o mapa mental do usuário e ajudando sua orientação. Para a implementação dos browsers propostos, foram utilizadas algumas rotinas oferecidas pelo editor de grafos compostos D-ABDUCTOR [Misu94]. Essas rotinas forneceram o algoritmo para layout automático de grafos e os recursos de animação.

Para ilustrar melhor o algoritmo implementado pelo browser de hiperbase, as Figuras 2 e 3 apresentam um exemplo de visões olho-de-peixe para a hiperbase pública do HyperProp [SoCR95], contendo um nó de contexto do usuário (nó de composição) *A* (toda a janela), que representa uma pequena parte de um relatório sobre o Carnaval Brasileiro. As relações de inclusão em nós de composição são representadas pela inclusão de retângulos (nós) em outros retângulos (nós de composicão). As figuras mostram como o mapa é modificado dependendo do nó em foco e do nível de detalhe escolhido pelo usuário.

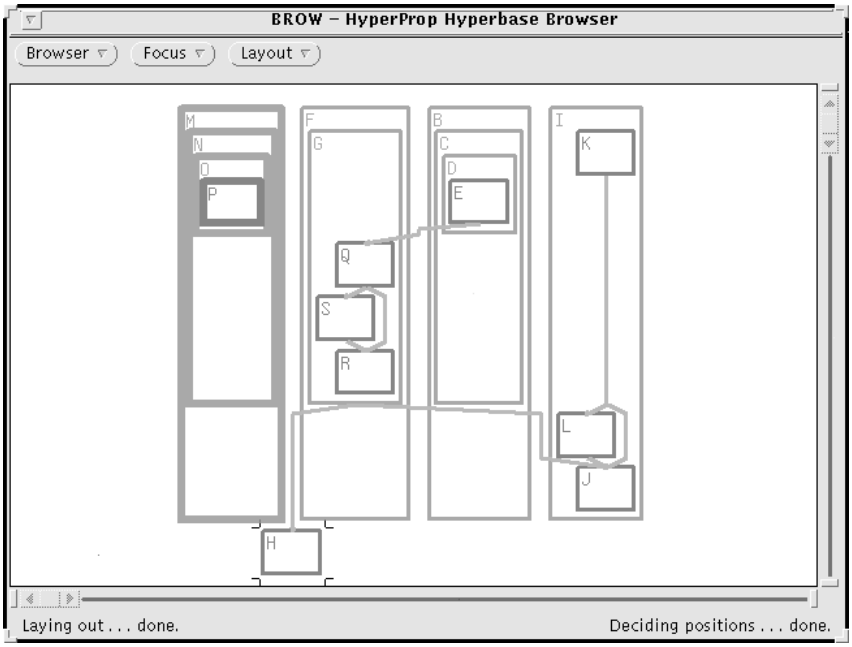
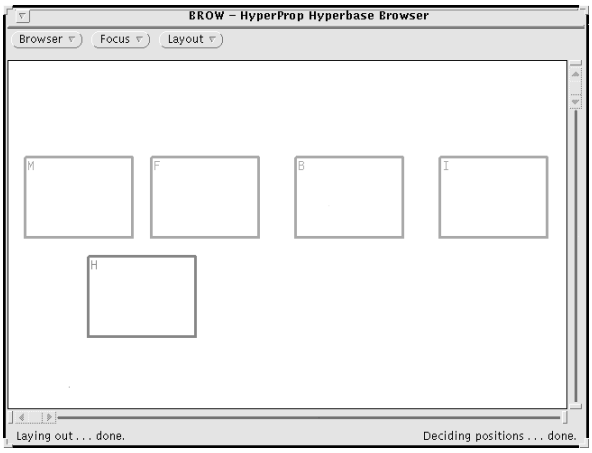
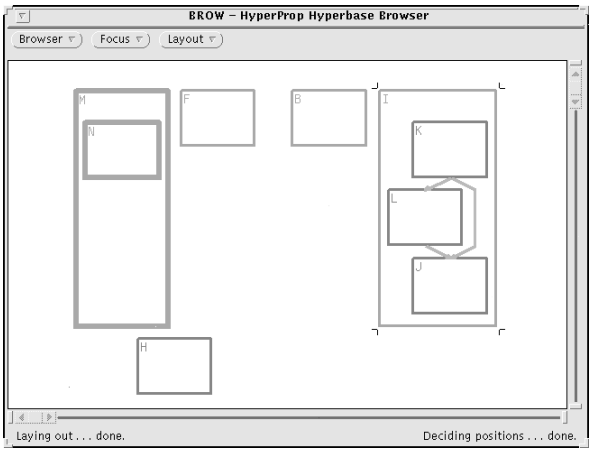


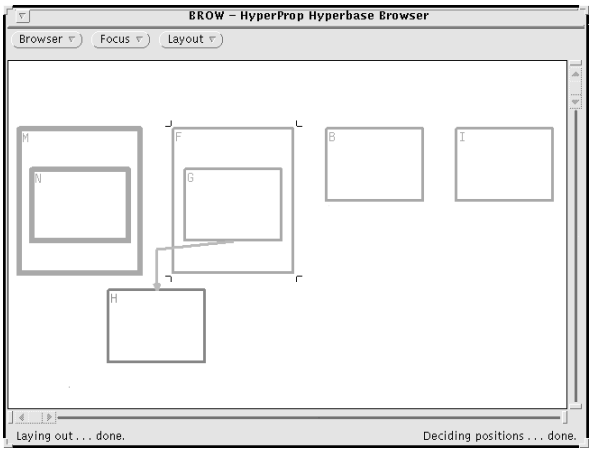
Figura 2 - Visão detalhada de todos os componentes aninhados do nó de contexto de usuário *A*. *<M,N,O,P>* é um landmark.



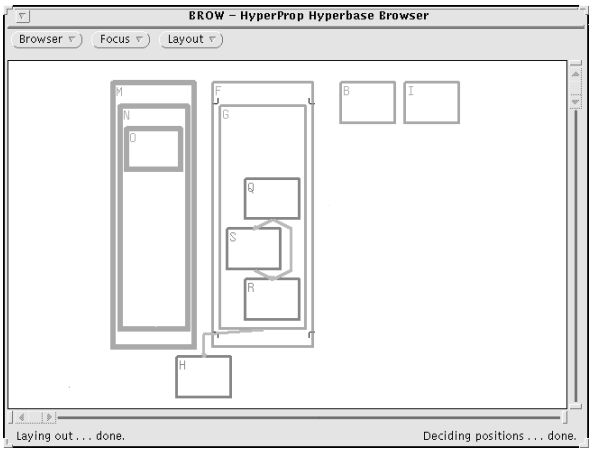
a) Visão inicial quando o browser abre o nó *A*



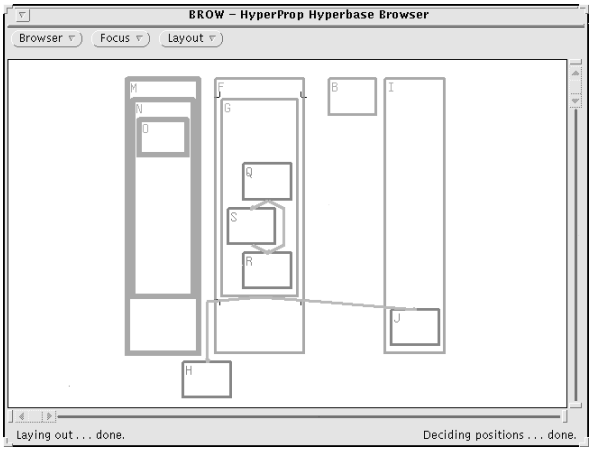
b) Foco no nó *I, DOI* ≥ *-4* (todos os nós exibidos têm *DOI* ≥ *-4*, valor definido pelo usuário)



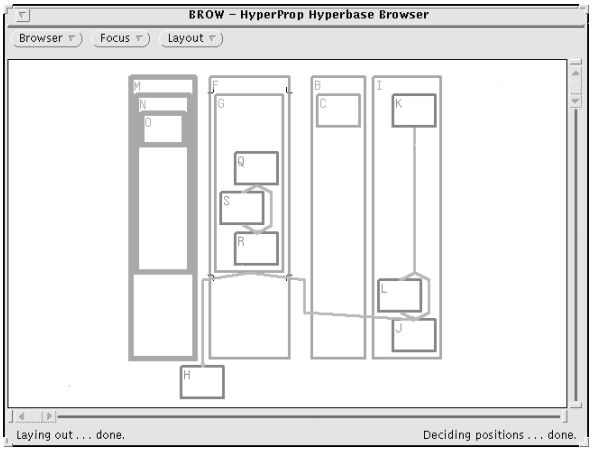
c) Foco no nó *F, DOI* ≥ *-4*



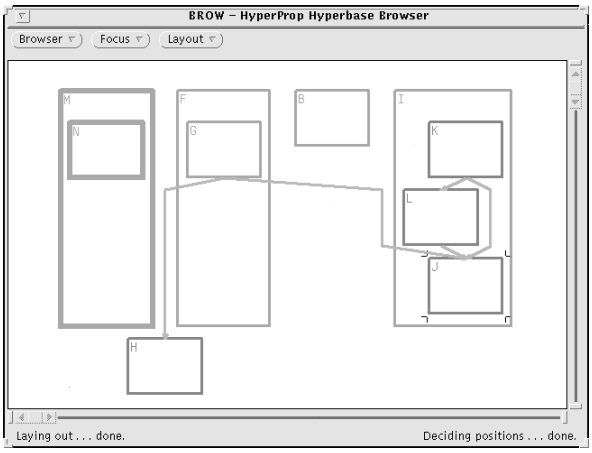
d) Foco no nó *G, DOI* ≥ *-4*



e) Foco no nó *G, DOI* ≥ *-5* (com mais detalhes)



f) Foco no nó *G, DOI* ≥ *-7* (com mais detalhes ainda)



g) Foco no nó *J, DOI ≥ -5*

Figura 3 - Exemplo de visões olho-de-peixe do nó de contexto de usuário *A* no browser de hiperbase do HyperProp

4 Trabalhos Relacionados

Construir editores gráficos para estruturas de grafo complexas não é uma tarefa fácil quando não podemos apresentar apenas uma visão ponto-a-ponto da estrutura mostrando todos os seus vértices e arcos. Uma maneira de resolver esse problema foi proposta no CYBERMAP [Gloo91], reduzindo o número de vértices exibidos aos usuários através do agrupamento. O sistema agrupa vértices em estruturas chamadas HyperDrawers, analisando o seu conteúdo e suas características, desconsiderando os arcos da estrutura. Depois de definir os HyperDrawers, o sistema apresenta uma visão ponto-a-ponto dessas estruturas dinâmicas. Esse editor, entretanto, ignora a semântica presente nas relações modeladas por arcos e, normalmente, a criação dos HyperDrawers é complexa.

Em modelos de grafos com vértices de composição aninhados, esses grupos de vértices são dados pela própria estrutura de composições e pelos arcos existentes. Esse fato possibilita a construção de diagramas que não mostram necessariamente todos os vértices da estrutura, mas somente aqueles relacionados a um ponto de vista específico, dado pelo próprio modelo.

Na proposta apresentada neste artigo, sugerimos um diagrama que balanceia informações locais e globais, permitindo que o usuário escolha o nível de detalhe apresentado. A ferramenta de edição gráfica explora características de modelos com composições aninhadas para garantir a legibilidade dos mapas, explodindo ou implodindo composições.

A estratégia básica proposta por Furnas [Furn86] é aplicada a estruturas onde os componentes da função de grau de interesse podem ser facilmente definidos, por exemplo, listas e árvores. Outros autores propuseram algumas extensões à estratégia tentando resolver o problema para grafos genéricos [ToDi92, Gloo91]. Nosso trabalho estende o modelo de olho-de-peixe para estruturas que permitem relações de composições aninhadas, além da relação usual modelada por arcos.

Existem outros trabalhos [MiSu91, SaBr92, Noik93] que estendem o modelo de olho-de-peixe para composições aninhadas introduzindo algumas considerações de layout na função de grau de

interesse. Essas extensões à proposta de Furnas utilizam visões olho-de-peixe para modificar o tamanho e a posição dos vértices no mapa. A diferença básica entre a nossa proposta e os trabalhos descritos em [SaBr92, Noik93] é que eles não definem a componente distância da função de grau de interesse considerando a distância abstrata entre vértices. Em [SaBr92], a distância entre dois vértices corresponde à distância euclideana em um layout inicial, não tendo nenhuma relação com a estrutura de vértices e arcos. Em [Noik93], somente a hierarquia de composições é utilizada para computar a distância entre vértices, sem considerar os arcos definidos na estrutura. Esses trabalhos podem ser utilizados para melhorar o layout da visão olho-de-peixe. Nosso principal objetivo quando usamos a estratégia olho-de-peixe é filtrar as informações a serem exibidas ao usuário. É muito importante calcular a função de grau de interesse baseada na distância abstrata entre vértices considerando a posição do usuário (foco) na estrutura do grafo. Esse procedimento dá mais importância às informações relacionadas ao interesse do usuário (foco), permitindo que a visão olho-de-peixe mostre uma visão geral da estrutura do grafo e mais detalhes sobre a posição do usuário na estrutura.

O editor de grafos compostos D-ABUCTOR [Misu94] utiliza algumas técnicas para melhorar o layout dos grafos baseadas em lentes olho-de-peixe, descritas em [Misu91]. Entretanto, as funções utilizadas para o cálculo do valor de interesse do usuário não satisfazem os requisitos apresentados na Seção 2. Neste editor, a distância entre dois vértices em relação aos arcos da estrutura não considera os vértices de composição do caminho dos arcos. Por isso, quando um vértice é exibido, os supergrafos que o contêm não são necessariamente exibidos no diagrama.

5 Conclusões

Este artigo discutiu a construção de ferramentas gráficas para a exibição de estruturas de grafos que permitem subgrafos aninhados. Essas ferramentas devem sempre garantir a claridade dos diagramas e manter a legibilidade, permitindo que os usuários escolham o nível de detalhe de informações a ser exibido.

A solução proposta leva em consideração o fato que, em modelos com subgrafos aninhados, as composições geralmente estruturam os supergrafos. Então, é possível construir editores que explorem o recurso de explodir ou implodir subgrafos, mostrando ou escondendo seus componentes, dependendo do interesse corrente do usuário. Este recurso limita a quantidade de informação apresentada no diagrama, melhorando sua legibilidade.

Foram citados alguns exemplos de utilização das técnicas propostas, nas áreas de linguagens de programação, engenharia de software e sistemas hipermídia. Para realizar sua validação, foram implementados os browsers do sistema hipermídia HyperProp [SoCR95]. A implementação foi feita utilizando a linguagem C++ na plataforma SunOS. O algoritmo para layout automático de grafos usado para desenhar os diagramas foi descrito por Sugiyama e Misue em [SuMi91], a quem agradecemos por permitir que utilizássemos o código-fonte do sistema D-ABDUCTOR [Misu94].

Referências

[Carv96] S. Carvalho, Subsystems in 2GOOD, *Relatório Técnico sobre o Projeto ARTS - Subprojeto 2GOOD*, n. 4, 1996, em preparação

[Dula90] N. Dulay, A Configuration Language for Distributed Progamming, *Tese de Doutorado*, Imperial College of Science, Technology and Medicine, University of London, Fevereiro, 1990

[Furn86] G. Furnas, Generalized Fisheye Views, *Proceedings of CHI'86 Human Factors in Computing Systems*, Boston, Abril, pp. 16-23, 1986

[Gloo91] P. Gloor, CYBERMAP Yet Another Way of Navigating in Hyperspace, *Proceedings of the Third ACM Conference on Hypertext*, pp. 107-121, San Antonio, Texas, Dezembro, 1991

[Hala88] F. G. Halasz, Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems, *Communications of the ACM*, Vol. 31, n. 7, pp. 836-852, Abril, 1988

[MiSu91] K. Misue and K. Sugiyama, Multi-viewpoint Display Methods: Formulation and Application to Compound Graphs, *Proceedings of the Fourth Conference on Human-Computer Interaction*, Stuttgart, F.R. Germany, Setembro 1-6, 1991

[Misu94] K. Misue, D-ABDUCTOR 2.30 User Manual, Institute for Social Information Science, FUJITSU LABORATORIES LTD., Japan, 1994

[Much96] D. C. Muchaluat, Browsers e Trilhas para Documentos Hipermídia Baseados em Modelos com Composições Aninhadas, *Master Thesis*, Departamento de Informática, PUC - Rio, Brasil, Março, 1996

[Noik93] E. G. Noik, Exploring Large HyperDocuments: Fisheye Views of Nested Networks, *Proceedings of the 5th ACM Conference on Hypertext*, Seatle, Washington, USA, pp.192-205, Novembro, 1993

[PPBC96] Virginia C. C. de Paula and Frederico C. G. Pereira and Sílvio S. Bandeira and Paulo R. F. Cunha, DisCo: Um Ambiente para Programação Distribuída Dinamicamente Reconfigurável, *Simpósio Brasileiro de Linguagens de Programação*, Belo Horizonte, MG, Setembro, 1996 (artigo submetido)

[ReKr75] DeRemer and H. Kron, Programming in-the-large versus Programming in-the-small, *Proceedings of the Local Area Communications Network Symposium*, Maio, 1975

[SaBr92] M. Sarkar and M. Brown, Graphical Fisheye View of Graphs, *Proceedings of CHI'92 Human Factors in Computing Systems*, pp. 83-91, Maio, 1992

[SoCR95] L. F. G. Soares and M. A. Casanova and N. L. R. Rodriguez, "Nested Composite Nodes and Version Control in an Open Hypermedia System". *Information Systems, Special issue on Multimedia Information Systems*, Vol. 20, n. 6, pp.501-519, 1995

[SuMi91] K. Sugiyama and K. Misue, Visualization of Structural Information: Automatic Drawing of Compound Graphs, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, n. 4, pp. 876-892, Jullho/Agosto, 1991

[ToDi92] K. Tochtermann and G. Dittrich, Fishing for Clarity in Hyperdocuments with Enhanced Fisheye-Views, *ACM ECHT Conference*, pp. 213-221, Milano, Italy, Novembro, 1992