Gerenciamento de Objetos em Sistemas de Hipertexto: Uma Proposta

José Fernando Tepedino[1]
Centro de Processamento de Dados
Universidade de Brasília
Campus Universitário, Brasília - DF

Eduardo Simões de Albuquerque
Silvio Lemos Meira
Departamento de Informática
Universidade Federal de Pernambuco
CP 7851, 50739 Recife - PE - Brasil

Resumo

Apresentamos um sistema de gerenciamento de objetos para sistemas de hipertexto. O sistema gerenciador, desenvolvido em um ambiente orientado a objetos, é utilizado pelo sistema de hipertexto *Acqua*.

Abstract

An object manager for hypertext systems is introduced in this work. Developed in an object-oriented environment, the object manager is used by *Acqua* hypertext system.

1 Introdução

Sistemas de hipertexto possibilitam a representação de informações organizadas de forma não-linear. Eles se propõem a manipular qualquer tipo de informação digitalizável, sendo bibliotecas eletrônicas de conhecimento. Neles as informações estão dispostas na forma de um grafo dirigido, podendo uma fazer referência a diversas outras.

O conceito de hipertexto, segundo Conklin [Con87], é muito simples: temos basicamente janelas na tela de um computador associadas a objetos (contendo as informações) num banco de dados e ligações entre esses objetos, as quais são representadas tanto nas janelas, na forma de "*buttons*" correspondendo às origens das ligações, quanto no banco de dados. Assim, a ativação de uma ligação na janela de uma informação provoca a abertura de outra janela de informação.

Sistemas de hipertexto são essencialmente interativos, envolvendo a representação das informações dispostas na forma de um grafo dirigido e as operações sobre as mesmas, as quais são basicamente duas: a criação da rede de informações e suas ligações (autoria) e a ativação de ligações nas janelas através de um dispositivo de apontamento, como um *mouse* (leitura).

Um dos grandes problemas dos atuais sistemas de hipertexto está no mecanismo de gerência de objetos pois ao contrário dos banco de dados, sistemas de hipertexto lidam com informações de natureza diversa como som, texto, imagem, animação, programas executáveis, etc. De modo geral, a informação manipulada por tais sistemas não possui estrutura rígida, como é comum em bancos de dados e o tamanho de cada *unidade* de informação também é variável. Um nó que contém uma imagem em 256 cores pode ter dezenas de kilo bytes e uma seqüência de animação armazenada digitalmente pode ter alguns mega bytes.

Outro grande complicador para o armazenamento, recuperação e administração de objetos em sistemas de hipertexto está na quantidade e natureza das ligações existentes entre os diversos

[1] Atualmente no LiNES - Departamento de Informática - UFPE

nós que formam a rede de um *hiperdocumento*. Cada nó pode ter dezenas de ligações para outros e a origem de cada ligação pode ser um trecho de texto, uma região de imagem, um ícone etc. Além destes fatores, como qualquer aplicação de hipertexto é essencialmente interativa, o tempo de resposta do sistema tem que ser extremamente pequeno. Ninguém usaria um sistema de hipertexto que leva um minuto ou mesmo muitos segundos para ter suas referências ativadas.

Devido a estes problemas, ainda não foram encontradas soluções definitivas para o problema de armazenamento de objetos em sistemas de hipertexto. Tentativas de usar sistemas de banco de dados convencionais têm sido feitas mas com resultados não plenamente satisfatórios. Sistemas de sucesso como o Hyperties, utilizam arquivos convencionais para o armazenamento de *nós*, no lugar de gerenciadores de bancos de dados.

O objetivo deste trabalho é apresentar uma proposta de um sistema gerenciador de objetos para sistemas de hipertexto. Este sistema trata tanto a memória principal como o armazenamento em disco dos objetos de um sistema de hipertexto e está sendo testado no sistema *Acqua* [MATS91].

2 O Sistema *Acqua*

O sistema *Acqua*, em desenvolvimento pelo Grupo de Linguagens e Engenharia de Software do Departamento de Informática da UFPE é o sucessor de *H* [Alb89, MAT90], totalmente implementado em Smalltalk-V/286 [Dig88].

Acqua possui várias características que não foram implementadas em *H*, que era um sistema experimental. Em *H* as idéias, mesmo não consolidadas, eram implantadas e sua avaliação era feita a posteriori. Normalmente uma característica era programada e ajustada de acordo com o resultado obtido, sem haver qualquer definição, mesmo informal, de como deveria ocorrer a implementação. Como conseqüência disso, desenvolvemos um sistema com diversas características —algumas bastante sofisticadas— mas que sobrecarregavam demasiadamente a interface com funções que o usuário normal não necessitava.

Em *Acqua* tivemos um pré-projeto de todo o sistema, com a parte central especificada formalmente em VDM [Jon86] e os *serviços*, desenvolvidos por grupos separados, informalmente.

Dentre o que julgávamos os maiores problemas de *H* e pretendíamos corrigir em *Acqua*, citamos:

- Interface: era baseada na interface do próprio Smalltalk-V, multijanelas com menus do tipo *pop-up*, mas sem o *look* Windows que hoje é considerado padrão. Em *Acqua*, nosso objetivo era implementar uma interface mais próxima do padrão Windows, mais simples e *auto-explicativa* do que aquela que conseguimos em *H*, de forma que certas operações mais usuais pudessem ser facilmente visualisadas.

- Navegação: implementar mecanismos mais sofisticados de navegação, incluindo possibilidade de navegação "por estrutura", além dos mecanismos de navegação por conteúdo já existentes em *H*. Navegação "por conteúdo" é aquela que considera apenas a informação dentro dos nós, sem considerar como estes se posicionam na rede que forma o documento. Como exemplo de navegação por conteúdo temos a busca de nós de informação por *strings* e a localização a partir de palavras-chaves associadas aos nós.
  A navegação por estrutura considera a rede que forma o documento e não o conteúdo de cada nó individualmente. Um exemplo de busca que usa os dois mecanismos seria

  > localize os nós com pelo menos duas ligações saindo e que contenham a palavra "hipertexto".

  Para reduzir a desorientação queríamos também fornecer "mapas" de modo que o usuário pudesse visualizar a rede de nós graficamente e ativar qualquer nó a partir do mapa.

- Especificação: como dissemos, *H* foi especificado parcial e informalmente. Um dos objetivos do projeto de *Acqua* era também testar o projeto e desenvolvimento orientado a objetos de porte relativamente grande. Como *Acqua* deveria ser desenvolvido por grupos trabalhando isoladamente, a necessidade de uma especificação bastante completa do núcleo do sistema era essencial. O modelo de dados H_2O foi especificado formalmente e a sua especificação teve a função tanto de documentar o sistema como de servir como referência para a implementação dos demais módulos do sistema *Acqua*: Interface, Navegação e Segurança.

- Segurança: em *H* não havia distinção entre *autores* e *leitores* —todo usuário tinha acesso a todas as funções e documentos do sistema para leitura e escrita—. Esta característica era interessante em um sistema experimental por estimular os usuários a testar todas as potencialidades do sistema. Por outro lado, qualquer um podia interferir no trabalho de outros.
  Para um sistema de uso prático é essencial que leitores sejam separados de autores. O normal é a existência de muitos leitores para cada autor. É também essencial que os usuários tenham uma visão limitada da base de documentos, só tomando conhecimento dos objetos aos quais têm acesso. Além disso, é interessante *ver* a base de documentos de acordo com assuntos ou tópicos, de modo a facilitar a escolha dos documentos.
  Apesar da separação feita entre *leirores* e *autores* feita no sistema, existe em *Acqua* a possibilidade de um leitor fazer comentários (chamados *anotações*) sobre nós de informação, mas sem alterá-los diretamente; temos ainda uma visão da biblioteca do sistema (chamada *Biblioteca Pública*) onde qualquer usuário pode ler certos documentos e neles fazer anotações.

- Gerenciamento de Objetos: em *H*, todos os objetos eram mantidos dentro da imagem de Smalltalk, em memória principal. Conseqüentemente, qualquer aplicação estava limitada à memória da máquina que executava o sistema Smalltalk. Desta forma, aplicações reais estavam descartadas. Em *Acqua*, o próprio sistema deveria gerenciar seus objetos de modo a permitir a existência de aplicações de qualquer tamanho.
  Como Smalltalk trabalha com objetos sempre em memória principal, tivemos que considerar no gerenciador de objetos diversos fatores relacionados com o ambiente Smalltalk, como a representação de objetos em disco e a transferência dos mesmos entre as memórias principal e secundária.

Todo o projeto de *Acqua* foi baseado nestas considerações. Na próxima secção fazemos uma descrição sumária do sistema e uma mais completa do sistema e seu processo de desenvolvimento pode ser encontrada em [MATS91, Mar91].

2.1 O Modelo de Dados de *Acqua* (H_2O)

O núcleo de *Acqua* é o modelo de dados H_2O, projetado para funcionar como uma *Máquina Abstrata de Hipertexto* [CG88] para *Acqua*. O modelo foi especificado formalmente em VDM [Jon86] e todos os *serviços* implementados sobre o mesmo, que fornece apenas os serviços básicos do sistema. O modelo é hierárquico e possui a seguinte estrutura:

Biblioteca
  Estante
    Documento (Documento Simples ou com Versões, Carta e Anotação)
      Nó (Nó simples ou Pilha de nós)
        Versão de Nó.

Todos os documentos do sistema estão em uma *Biblioteca* única que é dividida em *Estantes* de *Documentos*, formados por *Nós* que podem ter *Versões*, agrupadas em estruturas na forma de pilha.

Apesar de *Acqua* possuir apenas uma Biblioteca, o modelo de dados H_2O não faz restrição à quantidade de bibliotecas, tornando fácil a implementação de futuras expansões.

2.2 Características Gerais de *Acqua*

Dentre as principais características de *Acqua* destacamos:

- Nós: *Acqua* admite três tipos de nós: texto *comum*, código Smalltalk e *graphics*. O código Smalltalk é editado como um texto normal ou compilado e executado dependendo do modo como o documento é percorrido. Um nó do tipo *graphics* pode ter imagens preto e branco ou coloridas com até dezesseis cores.

- Referências: Qualquer tipo de nó pode conter referências a outros nós. No caso de um nó de texto ou código a origem da referência é uma *string* e no caso de uma imagem um retângulo dentro da figura. O destino de uma referência é sempre um nó.
  As referências podem ser *simples* ou *composta*, sendo a primeira direta a um nó e a segunda associada a uma lista de alternativas, cada uma correspondendo a outra referência (simples ou composta). Na interface do sistema, ao ser ativada uma referência simples, a janela do nó destino é aberta imediatamente para o usuário, enquanto que um *menu* de opções é apresentado na ativação de uma referência composta e a seleção de uma opção implicará na ativação de uma outra referência.
  O sistema não faz nenhuma restrição quanto ao nível de *menus* numa referência *composta*, de forma que ele depende exclusivamente do autor do documento.

- Documentos: Um documento é formado por um conjunto de nós associados por uma rede que é um grafo orientado. Os documentos são de dois tipos principais, que podem ser definidos pelo autor no momento de sua criação:

  1. Documento Simples: aqui não é mantido controle de versões sobre as alterações sofridas durante a vida do documento. Como exemplo de documentos simples temos *hipercartas* e *anotações*.
  2. Documento com Versão: *Acqua* mantém um controle cronológico sobre as alterações sofridas pelos nós. A navegação é sempre feita na versão mais recente do nó em questão, embora o sistema permita a navegação por versões antigas do documento.

- Biblioteca: Todos os documentos de *Acqua* estão em uma biblioteca única, dividida em estantes, que por sua vez contém os documentos. Por possuir um modelo de dados hierárquico, *Acqua* não permite o compartilhamento de nós entre documentos e nem de documentos entre estantes.

- Visões: *Acqua* supõe que todo documento possui uma estrutura linear básica. A quebra desta estrutura é feita pelas referências simples e compostas. Uma visão afeta apenas a estrutura linear do documento. Seu funcionamento é similar ao de uma referência composta, com a diferença que o leitor pode definir em qual visão deseja navegar o documento e a partir deste instante o sistema funcionará como se o documento tivesse sido escrito para apenas aquela opção, escondendo do leitor a existência das demais visões.

- Leitores e Autores: Em *Acqua autores* possuem status diferente de *leitores*. Esta característica é essencial para sistemas práticos, pois, de modo geral, o desejável é ter um conjunto de autores (produtores) de *hiperdocumentos* e um grande número de leitores (consumidores).

O sistema também permite que um leitor, embora não altere o documento em si, crie anotações que funcionam da mesma forma que notas deixadas às margens de livros.

- Grupos de Trabalho: A visibilidade sobre todo sistema é atribuída a *grupos de trabalho*. Assim, para utilizar *Acqua* o usuário deve estar ligado a um grupo de trabalho, onde é estabelecido o nível de segurança e visão da biblioteca.
  A definição de visões parciais da biblioteca do sistema e de autorizações de acesso aos documentos tem influência no gerenciamento de objetos, uma vez que restringem o escopo de trabalho dentro de uma sessão do usuário e definem que informações são passíveis de serem alteradas, sendo que este último caso pode gerar duplicação de uma informação em memória enquanto ela estiver sofrendo alteração.

- Sessão do Usuário: Toda interação do usuário com *Acqua* é feita dentro de uma sessão, relacionada a um grupo de trabalho na medida em que deve manter controle sobre o acesso ao sistema. Um usuário pode manter mais de uma sessão em aberto, desde que pertença a mais de um grupo.

- Correio Eletrônico: Para facilitar o trabalho cooperativo, *Acqua* mantém um serviço de correio eletrônico, onde uma *carta* é um *hiperdocumento* e pode ter referências para quaisquer outros documentos mantidos pelo sistema.

- Mapas de Navegação: Para facilitar a orientação dentro do sistema, *Acqua* permite que seja montado um mapa com toda a rede que forma um documento. Uma *calculadora* aclopada ao mapa permite que sejam verificados graficamente os nós que satisfazem a determinadas condições de conteúdo e estrutura. O sistema também mantém uma *bússola*, de modo que um leitor tem uma idéia de *norte* durante a sessão, facilitando sua orientação no *hiperespaço*.

- Caminhos de Navegação: *Acqua* permite que sejam armazenados caminhos pré-definidos para uso posterior. Um caminho pode ser, por exemplo, o roteiro mínimo de uma aula. O aluno deve seguir pelo menos os nós definidos pelo professor, mas nada impede que durante o processo de navegação ele visite outros nós, com o sistema se encarregando de manter o controle deste caminho, de modo que independentemente dos desvios que faça é sempre possível voltar ao caminho original.

3 Dificuldades de Gerenciar Objetos em Sistemas de Hipertexto

Hiperdocumentos consistem basicamente de nós de informação a partir dos quais saem referências a outros nós que também podem ter referências a outros, e assim por diante, formando um grafo dirigido. Algumas vezes, os nós possuem características especiais, de acordo com o tipo de informação contida, e podem estar agrupados de diversos modos, como, por exemplo, na hierarquia *"Biblioteca–Estante–Documento–Nó–Versão de Nó"* do sistema *Acqua*. Apesar disso, a estrutura básica de nós e referências continua válida, pois podemos considerar esses tipos de agrupamento apenas como maneiras de se organizar as informações.

Podemos então considerar o grafo de nós de informação ligados por referências como o aspecto "estático" de um sistema de hipertexto: isto não quer dizer que alterações não ocorrem, mas sim que a estrutura existe durante um certo período de tempo e sofre um número relativamente pequeno de mudanças.

É sobre o grafo de hipertexto que diversas operações são feitas, em especial por um *leitor*, que acessa informações de nós e escolhe referências para novas informações. Enxergamos este (a navegação pelo grafo de hipertexto) como sendo o aspecto "dinâmico" de sistemas de hipertexto,

especialmente por sua imprevisibilidade. Podemos ter no sistema caminhos pré-definidos ou caminhos mais freqüentemente utilizados à disposição dos usuários, mas estes têm a liberdade de seguir uma referência qualquer a partir de um nó de informação (saindo então do caminho pré-estabelecido). Assim, podemos saber a que nós um usuário pode se dirigir mas não o caminho exato a ser percorrido.

Surgem então, tanto em relação ao aspecto "estático" quanto ao "dinâmico", os problemas de armazenamento: qualquer critério de agrupamento de nós escolhido num sistema de hipertexto favorecerá sempre um certo (ou nenhum) caminho de navegação e prejudicará diversos outros. Visto que podemos ter diversos nós apontando para um certo nó e até mesmo referências entre nós de diferentes documentos, os caminhos percorridos pelos nós de um *hiperdocumento* podem ser os mais diversos, variando ao longo do tempo, ficando assim muito difícil estabelecermos um critério de agrupamento de nós. Uma idéia é agrupar ao máximo um nó e aqueles para os quais ele faz referência, o que poderia ser feito a partir da estrutura linear básica do documento —normalmente encontrada em um *hiperdocumento Acqua*— ou então ter um *hiperdocumento* "inteligente", capaz de efetuar estatísticas sobre o uso de seus nós e escolher o caminho de navegação mais freqüentemente utilizado para então adaptar o armazenamento dos seus nós de forma a favorecer esse caminho escolhido. Esse remanejamento dos nós armazenados em disco poderia tanto ser feito estaticamente, como um procedimento de reconfiguração do sistema, como dinamicamente, onde a disposição dos nós poderia ser alterada até mesmo durante a navegação de um *hiperdocumento* numa sessão *Acqua*.

4 Administração de Objetos H_2O em *Acqua*

Levando em consideração os problemas levantados na secção 3 e que os mecanismos de alocação de áreas e gerenciamento de arquivos em disco do sistema operacional sob o qual o sistema estiver sendo executado podem afetar o agrupamento de objetos, escolhemos tratar os elementos um a um, sem grande preocupação inicial com eficiência.

4.1 Considerações do Projeto

Alguns elementos sobre o gerenciamento de objetos em sistemas de hipertexto foram descritos em [TAM90]. Eles foram considerados na implementação do gerenciador de objetos para o sistema *Acqua*. Vejamos alguns pontos abordados:

4.1.1 Características

O sistema proposto deve gerenciar um grande volume de informação de natureza bastante variada, com muitos usuários, possivelmente trabalhando em estações conectadas em rede. O gerenciador se propõe a armazenar os objetos independentemente do ambiente em que se encontre a aplicação, sendo importante abordar os seguintes tópicos, comumente encontrados em sistemas de banco de dados:

- persistência das informações;
- consistência, em qualquer tempo, tanto das informações armazenadas quanto das estruturas de armazenamento e acesso;
- controle e autorização para acesso às informações (num ambiente multi-usuário e em rede);
- garantia da existência permanente de um espaço livre mínimo na memória principal;
- gerenciamento de objetos na memória principal e sua transferência entre a mesma e a memória secundária.

Persistência das Informações

Toda informação criada de forma persistente na aplicação precisa estar armazenada de forma duradoura, pelo menos durante sua *vida útil*. Dessa forma, toda alteração não-temporária feita em memória principal tem que ser espelhada na memória secundária.

Nem sempre é simples copiar objetos, pois podem existir referências a outros objetos e assim por diante. Tomando por base o sistema Smalltalk, encontramos em geral em ambientes orientados a objetos um controle automático de alocação de memória. Buscamos um mecanismo que não interfira diretamente no mecanismo de controle interno (se existir) do ambiente orientado a objetos no qual se encontre a aplicação.

Precisamos assegurar que seja feita uma cópia fiel das alterações, sempre que elas ocorrerem. Algumas decisões precisam ser tomadas em relação ao momento mais adequado para salvar uma alteração feita em memória.

Consistência

Mesmo nos piores casos, como a queda do sistema em meio a uma operação de atualização de informações, deve-se ser capaz de manter (ou restabelecer) sempre um estado válido, tanto para as informações da aplicação quanto em relação às estruturas internas.

Foi criado um mecanismo simples de armazenamento de informações em disco, na forma de listas encadeadas, mas bastante confiável, no qual as alterações são feitas numa área desocupada e efetivadas em uma única operação de gravação, de forma que qualquer interrupção faz com que as estruturas retornem ao estado anterior, ocorrendo no pior caso apenas a perda da última alteração.

Controle e Autorização de Acesso às Informações

Em um sistema multi-usuário (e conectado em rede) torna-se necessário termos tipos diferentes de usuário, com diferentes tipos de acesso e visibilidade de informações, e ainda mecanismos onde o autor ou responsável por um certo conjunto de informações estabeleça quem terá direito de acesso ou alteração.

Sessões dos usuários dentro de grupos de trabalho, visam tanto disciplinar o uso do sistema, minimizando interferências entre usuários, como facilitar o acesso aos documentos, através de visões parciais da biblioteca para diferentes grupos de trabalho.

Acesso concorrente por processos paralelos numa mesma estação ou ligados via rede, mesmo com autorização, podem entrar em conflito quando há alterações em curso. Um mecanismo de controle e notificação de conflitos precisa ser definido.

Espaço Livre na Memória Principal

É preciso ter controle sobre a criação de objetos na memória principal, bem como sobre todas as referências feitas por objetos da aplicação. Assim, verificamos se a memória de trabalho disponível é suficiente para a criação e manipulação de novos objetos ou se é necessário transportar alguns deles para a memória secundária, liberando dessa forma área para trabalho. O controle de referências é essencial para objetos retirados da memória, pois a detecção de uma tentativa de acesso fará com que o gerenciador de objetos traga de volta um objeto transferido para disco.

Deve-se ter uma política de retirada de objetos, de acordo com a freqüência de acesso e o tamanho dos mesmos, onde se considere alguma diferenciação entre (certos) tipos de objetos: alguns deverão sempre permanecer na memória principal, outros são passíveis de transferência para disco, enquanto outros deverão permanecer sempre em disco (sendo apenas temporários na memória principal).

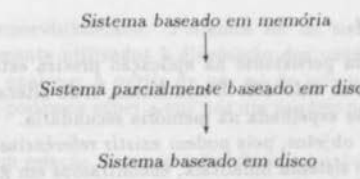

Figura 1: **Implementação em etapas.**

Na maioria dos casos a transferência de objetos dar-se-á apenas do disco para a memória, pois os objetos em memória são sempre espelhados em disco.

A permanência de objetos na memória é feita de acordo com ações ativadas pelo usuário, como, por exemplo, seleção de uma referência na janela de um nó e abertura, ativação e fechamento de janelas.

Paginação e Transformação de Objetos

Deve haver uma representação e um método de transformação de objetos da memória principal em cadeias de caracteres na memória secundária. Tal processo deve ser rápido e eficiente nos dois sentidos e a representação em disco bastante compacta.

4.2 Metodologia de Implementação

Implementamos o gerenciador de objetos de *Acqua* em etapas, partindo da configuração inicial do sistema, totalmente baseada em memória principal, para uma totalmente baseada em disco, como mostrado na figura 1.

Utilizamos como configuração-base a primeira versão de *Acqua*, apresentada na "European Conference on Hypertext" (ECHT'90) em Paris em Novembro de 1990, onde todos os objetos existiam sempre em memória principal, dentro da imagem do Smalltalk. Era possível gravar e ler documentos inteiros em disco, mas os objetos tinham que estar em memória para serem manipulados.

O gerenciador de objetos foi dividido em dois módulos principais:

Módulo	*Responsabilidade*
• Controlador de nós e janelas	Controle de objetos em memória principal; Gerenciamento do espaço de trabalho em memória; Controle dos pontos de ativação (acessos a objetos);
• Gerenciador de objetos em disco	Transferência de Objetos entre o disco e a memória; Controle de objetos e estruturas de armazenamento em disco.

Etapas da Implementação

A implementação do gerenciador de objetos foi dividida em três grandes etapas:

1. **Sistema baseado em memória:** foram efetuadas as adaptações mínimas necessárias ao sistema *Acqua* para que as novas classes do gerenciador de objetos pudessem ser criadas. Foram também criados métodos para montar a descrição parcial de alguns objetos, como, por exemplo, uma estante sem os seus documentos;
2. **Sistema parcialmente baseado em disco:** o gerenciador de objetos em disco foi desenvolvido independentemente do controle de nós e janelas:

- Gerenciador de objetos em disco: cria um tipo especial de documento que pode ter nós tanto em memória quanto em disco, mas cujo protocolo (mensagens tratadas) continua o mesmo. Os nós são trazidos quando necessário, podendo ser removidos da memória. A criação e alteração de nós é feita na memória e sempre espelhada em disco;
- Controlador de nós e janelas: simula o controle de nós e janelas na memória principal. Este controle é o ponto-chave para o gerenciamento de nós, documentos e demais objetos no sistema. Controlar janelas abertas e a memória ocupada, detectando os pontos de ativação acionados pelo usuário na interface; faz o controle de acordo com critérios definidos pela política de gerenciamento (a qual é parametrizada, podendo ser modificada).

Por fim, os dois módulos acima, foram aclopados de forma que o controlador de nós e janelas trata nós de documentos tanto em memória quanto em disco, de acordo com a política de gerenciamento. Apenas nós de documentos em disco podem ser removidos da memória;

3. **Sistema totalmente baseado em disco:** na etapa final, todos os objetos do modelo de dados —biblioteca, estantes, documentos (simples e com versões), nós (e também pilhas de nós e suas versões), rotas, cartas, anotações, etc.— são armazenados em disco. A imagem do Smalltalk/V passa a servir apenas como plataforma para execução do sistema e não mais para armazenamento de dados.

4.3 Módulos do Gerenciador

O gerenciador de objetos é contituído de dois módulos principais: o controlador de nós e janelas e o gerenciador de objetos em disco como mostrado na figura 2 abaixo, onde é mostrado o acoplamento entre os módulos do sistema *Acqua*:

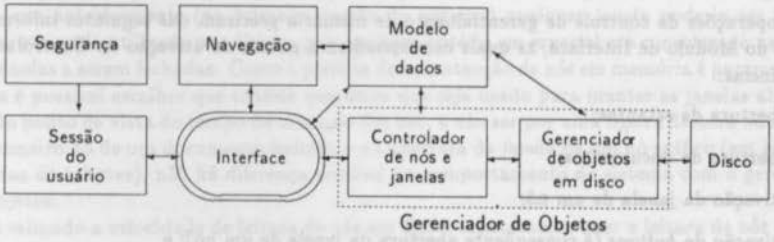

Figura 2: **Módulos do sistema Acqua.**

4.3.1 O Controlador de Nós e Janelas

Comunicando-se diretamente com o módulo de interface e com alguns elementos do modelo de dados (as estruturas que armazenam nós dentro dos documentos), o controlador de nós e janelas acompanha as ações efetuadas pelo usuário do sistema de forma a tornar o serviço mais eficiente, trazendo para a memória os nós cuja probabilidade de uso num futuro próximo é maior, solicitando-os ao gerenciador de objetos em disco. Temos os seguintes pontos-chave no sistema que são considerados para a ativação deste módulo:

- **Interface:** ao ser iniciada a sessão, o sistema busca os parâmetros (visão da biblioteca, autorização e tipos de acesso) definidos para o usuário dentro do grupo de trabalho selecionado, estabelecendo a devida visão da *hiper-biblioteca*;

- **Navegação:** nenhum ponto de ativação direto precisa ser considerado para este módulo, pois eles envolvem apenas o uso dos módulos de Interface e Modelo de Dados. Como a adição de novas características ao sistema é feita de maneira ortogonal às já existentes (o protocolo original de comunicação entre os objetos é mantido), nenhuma alteração se fez necessária no módulo de navegação, visto que os nós são fornecidos à medida que forem solicitados ao modelo de dados, estejam eles em disco ou em memória;

- **Modelo de dados:** busca de nós nos documentos. Uma estrutura de dicionário, com a identificação dos nós como chave, está definida dentro de um documento. Foi criada uma estrutura em memória que pode armazenar *parcialmente* os nós de um documento, solicitando-os ao gerenciador de objetos quando necessário, de forma transparente para o restante do sistema.

  Uma estrutura especial para o dicionário de nós de um documento em disco é construída, contendo uma parte dos nós do documento em memória e tendo como extensão o arquivo do documento. Assim, podemos detectar qualquer tentativa de acesso ou alteração a um nó, de forma a trazê-lo para a memória quando necessário e espelhar uma alteração feita em memória para o disco.

Este módulo controla os nós que se encontram na memória do sistema. Eles são trazidos do disco à medida que se tornam necessários e muitas vezes possuem uma janela associada. A carga de nós na memória tem por base a política de gerenciamento e remanejamentos são feitos, se necessário. Há uma lista de *nós carregados* responsável pela geração de informações sobre os nós presentes na memória principal, de acordo com a política de gerenciamento, respondendo a mensagens como:

"qual é o espaço total ocupado pelos nós" e

"retorne a lista de nós carregados ordenada (de diversas formas)".

As operações de controle de gerenciamento de memória precisam das seguintes informações obtidas do Módulo de Interface, as quais correspondem a pontos de ativação do Controlador de nós e janelas:

- abertura de estantes;
- abertura de documentos;
- ativação da janela de um nó;
- ativação de *buttons* (e conseqüente abertura da janela de um nó); e
- fechamento de janelas.

4.3.2 O Gerenciador de Objetos em Disco

Este módulo é responsável pela transferência e conversão de representação de objetos entre as memórias principal e secundária, e também pelo controle de acessos e alterações nas informações na *hiper-biblioteca*, armazenada em disco numa hierarquia de arquivos de documentos guardados em diretórios de estantes. Ele contém elementos de controle de abertura de arquivos e ainda os dicionários dos nós de documentos em disco, para permitir acesso direto aos mesmos.

A interface deste módulo com o modelo de dados é feita através de uma estrutura especial de armazenamento de nós em memória, um *Dicionário* Smalltalk, que se comunica com os demais módulos como se todos os nós do documento estivessem em memória, embora muitas vezes solicite ao gerenciador em disco a carga de um determinado nó do disco para memória antes de atender à solicitação.

4.3.3 Otimizações e o Ambiente

O método utilizado pelo sistema Smalltalk/V para representar qualquer objeto não recursivo fora da memória principal é o de construir uma *string* contendo uma seqüência de comandos que quando compilada e executada monta uma cópia do objeto original, o que possibilita a transferência de objetos entre diferentes imagens. Essa representação é obtida enviando-se a mensagem 'storeOn:" a um objeto. Estudos feitos no sistema nos mostraram que algumas representações geradas pelo sistema Smalltalk original eram bastante ineficientes, sendo um ponto crítico a representação de figuras, tratadas internamente na forma de "*bitmaps*".

Para tirar melhor proveito do ambiente Smalltalk, foi otimizado o método "storeOn:" para diversas classes elementares, tais como *Point, Rectangle, Bitmap*, coleções e outras, com o objetivo de reduzir a representação dos objetos do sistema. O método "storeOn:" otimizado gera representações compactas para figuras, guardadas na forma de "Bitmaps", originalmente demasiadamente extensas e ineficientes, tanto na geração quanto na sua recuperação. Como coleções são muito utilizadas em *Acqua* e em todo o ambiente Smalltalk, o método "storeOn:" foi adaptado para estes objetos.

Representações reduzidas são importantes, visto que ocupando um menor espaço em disco, os tempos de leitura, gravação e reconstituição dos objetos são também reduzidos, embora talvez em alguns poucos casos particulares o tempo de montagem possa ser um pouco maior do que seria utilizando os métodos originais do Smalltalk. Estudos estão sendo feitos na busca de uma representação ideal para objetos em disco; uma idéia seria guardar a seqüência de comandos que representa um objeto já compilada, reduzindo o tamanho da sua representação e tornando necessária apenas a execução (e não mais a compilação) do comando armazenado para a recuperação do objeto.

4.4 Avaliação

O controle de fechamento automático de janelas passou a ter um comportamento mais organizado, visto que anteriormente (na primeira versão do sistema) qualquer janela poderia ser fechada, inclusive aquela utilizada por último, pois nenhum critério em especial era considerado na escolha das janelas a serem fechadas. Como a política de manutenção de nós em memória é parametrizada, agora é possível escolher que critério queremos que seja usado para manter as janelas abertas.

Do ponto de vista do tempo de ativação dos nós, a não ser por uma ligeira demora na abertura do primeiro nó de um documento fechado e na abertura da janela de um nó gráfico (em geral com dezenas de k-bytes), não há diferença sensível no comportamento do sistema com o gerenciador de objetos.

Avaliando a velocidade de leitura de nós em disco, constatamos que: a leitura de nós de texto e de código é imediata; já a de um nó gráfico preto-e-branco é rápida; somente a de nós gráficos coloridos é que é um pouco demorada.

A gravação de nós em disco é um pouco mais demorada, mas as velocidades de gravação seguem a mesma ordem da leitura, como era de se esperar, inversamente proporcional aos tamanhos das representações dos objetos em disco: nós de texto e de código (mais rápidos), nós de figuras preto-e-branco e nós de figuras coloridas (mais demoradas). A tabela 1 mostra algumas medidas de tempo feitas do gerenciador de objetos, onde os tempos marcados se referem à conversão de objetos Smalltalk em *strings* e sua gravação em disco ou a leitura de *strings* e sua transformação de volta para objetos Smalltalk. Os valores mostrados na tabela 1 foram obtidos utilizando um processador 80386 com *clock* de 33MHz e um disco IDE de 18ms.

O remanejamento da memória ocorre com maior freqüência quando temos um grande número de nós gráficos carregados na memória, provocando uma breve parada do sistema antes da abertura da janela. Testes em uma máquina com memória principal de 8M *bytes* nos mostram que o remanejamento da memória tende a ocorrer com uma freqüência muito pequena.

tipo do nó	*leitura (seg)*		*gravação (seg)*	
	mínimo	máximo	mínimo	máximo
texto	0,110	0,170	0,220	0,600
código	0,160	0,220	0,220	0,600
gráfico preto-e-branco	0,710	1,920	1,700	3,500
gráfico colorido	1,970	12,630	5,710	12,790
pilha de nós	0,110	0,220	0,220	0,335

Tabela 1: Tempos médios (em segundos) de leitura e gravação de nós.

Uma observação a ser feita sobre as medidas de tempo mostradas acima é que elas costumam variar para um mesmo nó, de uma operação para outra, principalmente para nós gráficos. Esta variação ocorre mais na gravação do que na leitura, possivelmente devido à estrutura de armazenamento de nós em listas encadeadas nos arquivos dos documentos: como os registros não precisam ocupar necessariamente regiões contíguas, podem ocorrer configurações de mais rápido acesso do que outras, entre sucessivas gravações. Apesar disso, as medidas efetuadas são relativamente confiáveis desde que mantenhamos cada informação armazenada em regiões próximas umas das outras, visto que as medidas acima foram feitas em documentos de tamanho pequeno, com nós do tamanho das janelas da interface (cerca de 500 bytes para nós de texto e código e algumas dezenas de bytes para nós gráficos). Alguns utilitários de otimização de disco tendem a melhorar a performance geral do sistema.

Pretendemos refinar o esquema de armazenamento utilizado de forma a manter cada informação em áreas contíguas e fazer um tratamento especial para nós gráficos, cujas representações são bem maiores do que nós de texto.

5 Conclusão e Trabalhos Futuros

O sistema está implementado e em utilização em laboratório. Documentos de grande porte estão sendo escritos para testá-lo na prática. Não há indícios que o uso prático vá mostrar problemas não resolvidos pela atual implementação.

Em uma futura implementação o sistema terá um processo rodando em *background* carregando e descartando nós para/da memória principal (seguindo alguma heurística) para melhorar a performance. Por exemplo, quando um nó for ativado, este processo poderia imediatamente começar a carregar para memória os nós para os quais existem referências partindo deste. É provável que o próximo nó a ser ativado seja algum dos nós carregados. Da mesma forma, quando uma janela for fechada possivelmente diversos nós que estão carregados em memória deixam de ser *alcançáveis* diretamente a partir da configuração corrente da tela.

O sistema executa em uma única estação, ou com uma única base de dados em rede de estações. Numa base de dados distribuída teríamos mais de uma *Biblioteca* no sistema: poderíamos ter, por exemplo, uma em cada estação de trabalho. O modelo de dados *Acqua* pode ser facilmente estendido para que a identificação da biblioteca seja incorporada às identificações completas dos seus objetos. Precisaríamos então criar uma configuração em rede com diversos gerenciadores de objetos interligados, ao invés de um único, concentrado numa estação.

Um sistema *totalmente* baseado em disco terá um armazenamento em memória secundária que considera todos os demais elementos do sistema e não apenas os principais, de forma a tratar em disco sessões, usuários, grupos de trabalho e autorizações. Diversas idéias discutidas neste trabalho poderiam ser aproveitadas para o armazenamento desses elementos.

Agradecimentos

O primeiro autor deste artigo agradece ao Centro de Processamento de Dados da Universidade de Brasília (CPD-FUB), particularmente ao seu Diretor, Sr. Antônio Jorge Rachid, por ter possibilitado a realização deste trabalho junto ao Grupo de Linguagens e Engenharia de Software (LiNES) do Departamento de Informática da UFPE.

Referências

[Alb89] Eduardo Simões de Albuquerque. O Sistema de Hipertexto H. Master's thesis, Universidade Federal de Pernambuco, Recife - PE, 1989.

[CG88] Brad Campbell and Joseph M. Goodman. HAM: A General Purpose Hypertext Abstract Machine. *Communications of the ACM*, 31(7), Jul 1988.

[Con87] Jeff Conklin. A survey of hypertext. Technical Report STP-356-86, Microelectronics and Computer Technology Corporation, Austin, Tx, Feb 1987.

[Dig88] Digitalk. *Smalltalk-V 286 Tutorial and Programming Handbook*. Digitalk Inc., May 1988.

[Jon86] C. B. Jones. *Systematic Software Development Using VDM*. Prentice-Hall International, 1986.

[Mar91] José Fernando Tepedino Martins. Um Gerenciador de Objetos para Sistemas de Hipertexto. Master's thesis, Universidade Federal de Pernambuco, Recife - PE, 1991.

[MAT90] Silvio Meira, Eduardo Albuquerque, and José Fernando Tepedino. An Experience in Building an Object-Oriented Prototype of an Advanced Hypertext System. In *Anais do IV SBES*, São Paulo - SP, 1990.

[MATS91] Silvio Meira, Eduardo Albuquerque, José Fernando Tepedino, and Cássio Santos. An Experiment in Object-Oriented Design and Programming: The Acqua Hypertext System. Technical report, Universidade Federal de Pernambuco, Recife - PE, 1991.

[TAM90] José Fernando Tepedino, Eduardo Albuquerque, and Silvio Meira. Considerações Sobre o Gerenciamento de Objetos em Sistemas de Hipertexto (comunicação). In *Anais do IV SBES*, Águas de São Pedro - SP, 1990.