LINKS EM UM SISTEMA DE HIPERTEXTO

Alexandre M. L. de Vasconcelos, Ana Cristina V. de Melo e Silvio Lemos Meira

Departamento de Informática
Universidade Federal de Pernambuco
Cidade Universitária
50739, Recife - PE

RESUMO

Este artigo mostra a especificação formal de um Gerenciador de Links para Hipertexto. Tendo este tipo de sistema alto grau de complexidade, a especificação aqui relatada trata apenas de seu fundamento principal, os "links". Para isto, serão explicados inicialmente os princípios básicos de hipertexto, especificação formal, descrição informal das funções de manipulação de "links" e finalmente a especificação funcional do gerenciador.

1. INTRODUÇÃO

Um hipertexto é diferenciado de textos usuais porque armazena um texto estruturado de forma não linear. Para isto são necessários elementos adicionais, os "links" (ligações), que relacionam as suas unidades de informação.

O desempenho de um Sistema de Hipertexto está intrinsecamente relacionado com as operações que manipulam os seus objetos principais -os links. Por esta razão, faz-se necessário um gerenciador para tais operações.

A "idéia" de Hipertexto [1,2,4], ou pelo menos, de "construção" de um sistema de hipertexto é relativamente nova. Muitas sugestões são "dadas", mas nem todas implementadas. Deste modo, existe a necessidade da definição formal das "idéias" envolvidas para que possamos analisar a estrutura e a viabilidade dos seus elementos. Esta necessidade de especificação formal surge com a evolução do *software*, pois à medida que aumenta o seu grau de complexidade, é preciso ferramentas que dêem suporte aos projetistas na análise da integridade e coerência de seus sistemas.

Nas seções que se seguem, discute-se as principais características dos Sistemas de Hipertexto, uma metodologia para desenvolvimento de *software* e especificação do gerenciador de links.

2. CARACTERÍSTICAS DOS SISTEMAS DE HIPERTEXTO

A maioria dos sistemas atuais de edição de textos têem a capacidade de armazenar textos "únicos" (completamente independentes) e não interligáveis. Há algum tempo, surgiu a idéia de armazenar textos de forma não linear, assim como estão escritas as enciclopédias.

Na maioria das vezes, quando estamos lendo um determinado "assunto" em uma enciclopédia, é necessário consultar outros volumes à procura de explicações indicadas pelas referências do item pesquisado. Neste caso, seria cômodo se tivéssemos uma enciclopédia eletrônica, onde um "toque" em regiões sensitivas de uma tela nos levasse imediatamente à referência correspondente [1,3].

A partir da idéia de estruturar documentos de forma não linear armazenando-os em computador, surgiu o conceito de hipertexto (ou *hypermedia* [3]). Ao objeto armazenado nesta forma chamamos hiperdocumento[1]; nó é a unidade básica de informação; ponto, um átomo de informação; região, um conjunto de pontos; e links, as ligações existentes entre os nós dos hipertextos.

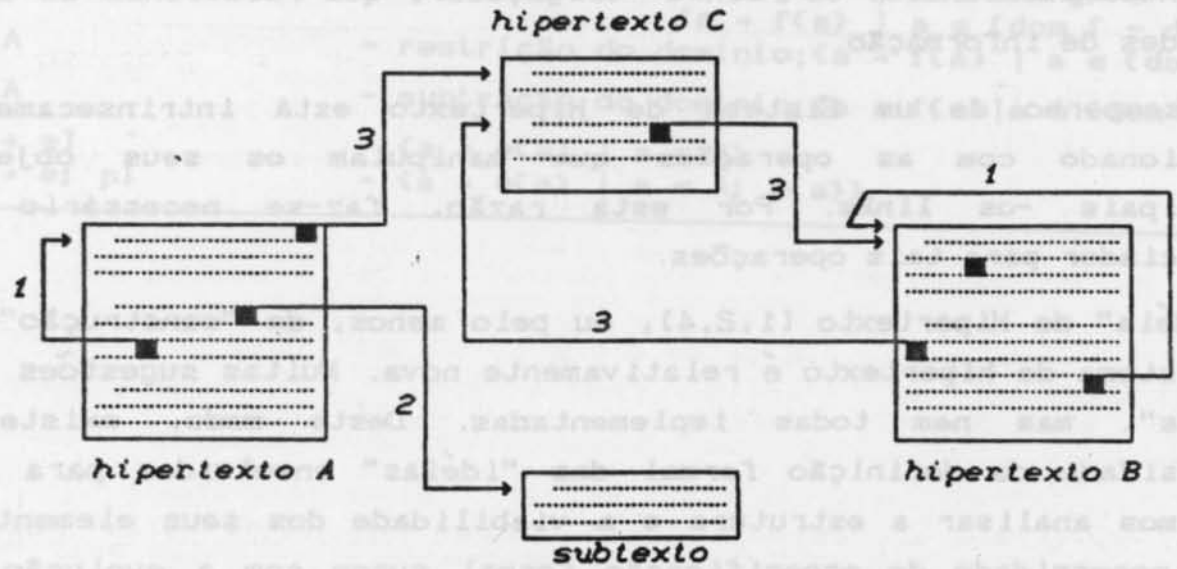

Figura 1 - Interligações de Hiperdocumentos

---

[1] Neste artigo, as palavras texto, documento, hipertexto e hiperdocumento são usadas com o mesmo significado, exceto no caso de texto/documento "usual".

Como pode-se ver na *Figura 1*, um link poderá ser usado para relacionar objetos em diferentes níveis [1,3]:

(1) Conectando um ponto ou região de um hiperdocumento a outro elemento dele próprio;

(2) Conectando um ponto ou região de hiperdocumento a um subtexto (comentário, anotação, nota de "rodapé", etc.), em geral invisível aos olhos do leitor;

(3) Conectando hiperdocumentos independentes.

Assim, nos casos (1) e (2) existe um link intra-documento, enquanto que no caso (3), inter-documentos.

Para que se possa "escrever" objetos (documentos) nesta forma, sejam eles textos, especificações, programas, etc, é necessário criar um sistema que administre todos os objetos e ligações. Na construção do sistema será necessário um editor de hiperdocumentos, um gerenciador de links, além dos gerenciadores de arquivo e visualização por exemplo. O "engenho" (gerenciador central) do Sistema de Hipertexto administra todas as funções do sistema, sejam elas de manipulação interna, ou comunicação com as *interfaces* que interagem diretamente com o usuário.

3. ESPECIFICAÇÃO FORMAL DE *SOFTWARE*

A necessidade, cada vez maior, de dotar o processo de desenvolvimento de *software* de técnicas e ferramentas que conduzam a uma maior disciplina e rigor fez surgir um estilo de programação que se aproxima das definições matemáticas.

A programação é apenas uma fase terminal do processo de desenvolvimento de *software*. Das várias fases que constituem este processo, a especificação é, como se sabe, uma fase fundamental do projeto, pois nela são definidas as principais características de constituição e comportamento do sistema [10,11].

Um erro durante a fase de especificação pode acarretar graves implicações no processo de desenvolvimento do *software*. Em geral, apenas perante o produto final o usuário percebe que este não realiza o pretendido ou então o faz de forma incompleta.

A utilização de métodos formais de especificação visa contornar estas deficiências e baixar os custos de desenvolvimento de

software. Assim, para que uma especificação possa refletir uma análise rigorosamente descrita do que se pretende que o sistema realize, e sobre que objetos, deverá ser feita numa linguagem com base matemática sólida, pois é muito freqüente gerar ambigüidades quando se especifica em uma linguagem natural.

4. A METODOLOGIA *METOO*

O método de especificação *metoo* [7,11] baseia-se no princípio: qualquer objeto abstrato do sistema pode ser modelado, ou representado, por um objeto matemático de propriedades bem definidas. A notação é próxima à usada nas linguagens de especificação VDM [5] e Z [12], sendo baseada numa combinação de programação funcional [9] e em uma parte construtiva da teoria dos conjuntos.

Devido às suas características de linguagem de muito *alto nível*, e até mesmo pelos símbolos de que faz uso, a linguagem de especificação abstrata da metodologia *metoo* não é diretamente interpretada de forma a gerar um programa. Na realidade, a construção de um protótipo será feita a partir da tradução da especificação abstrata para uma linguagem funcional semelhante a LISP [6] (*metoo* concreto), gerando-se assim um programa que poderá ser executado.

Cada objeto abstrato é representado usando-se as estruturas matemáticas de *metoo* (conjuntos, relações, funções finitas, tuplas e seqüências) [11], e as operações sobre estes objetos são as usuais em cada estrutura matemática. Tais operações são denotadas por funções matemáticas da forma $f: x \rightarrow y$, onde x é o tipo dos argumentos de f, e y é o tipo do resultado.

Uma vez escolhido o modelo matemático para representar os objetos, pode-se então iniciar a especificação do sistema, refinando-a sucessivamente até que se possa atingir a fase de construção de um modelo executável - o protótipo.

5. DEFINIÇÃO INFORMAL DO GERENCIADOR DE LINKS

Nesta versão inicial do sistema de hipertexto, consideramos a necessidade de armazenamento de textos tipo documento, especificação e programa. Para isso uma referência será denotada por uma palavra;

a unidade básica de referência é o parágrafo, ou seja, o usuário só poderá ter como referência mínima um parágrafo, ou então um conjunto de parágrafos; e a unidade mínima de visualização é a página.

Em geral, o parágrafo contém uma "idéia" e a quantidade de controle utilizada não é tão grande quanto no caso de termos palavra como unidade de referência. A página como unidade de visualização situa o parágrafo referido em um contexto.

Um hiperdocumento é uma seqüência de parágrafos, em conseqüência seu editor deverá ser "estruturado". Em vista disso, "assumimos" que o editor tem um reconhecimento de início, final e número do parágrafo na seqüência do texto, ou seja, um tratamento especial para a sua estrutura, o parágrafo, além de todos os outros reconhecimentos usuais de página, contexto de edição, etc.

O gerenciador de links de um sistema de hipertexto "comunica-se" diretamente com o editor, e faz o tratamento das operações realizadas sobre os links. Para isso, teremos uma série de funções:

- Criação de um link: estabelece um link entre o nó origem e o nó destino, levando em consideração o fato de existir, ou não, o nó destino.
- Remoção de um link: suprime o caminho de acesso entre o nó origem e o nó destino.
- Cópia de um link: faz com que o novo nó possua o mesmo caminho de acesso do nó origem.
- Transferência de um link: muda apenas a localidade do ponto de referência dentro do texto, preservando o caminho de acesso.
- Ativação de um link: em momento de edição, ou apenas leitura, "leva" o leitor ao nó destino previamente estabelecido.

6. ESPECIFICAÇÃO FORMAL DO GERENCIADOR DE LINKS

Como "passo" inicial do projeto do sistema de hipertexto, desenvolvemos a especificação formal e protótipo funcional em *metoo*. Assim, cada função de "tratamento" de links terá uma ou n funções correspondentes.

Definimos o modelo abstrato do sistema de links, ou seja, os objetos abstratos com os quais "trabalhamos":

Modelo Abstrato dos Objetos

```
BD           >> Base de Dados do sistema.
texto        >> Controle dos links de textos e subtextos
controle     >> Controle dos links referentes a parágrafos.
ref          >> Controle de uma referência.
tipo_ref     >> Tipos de referência.
subtxs       >> Controle de subtextos de um hiperdocumento.
pilha_naveg  >> Pilha dos pontos de aportamento na navegação
                de um link.
tabpag       >> Tabela que faz a correspondência entre as
                páginas e parágrafos de um hiperdocumento.
num_cria     >> Número de criação do parágrafo.
num_loc      >> número de localização do parágrafo.
```

Em termos gerais, a Base de Dados (BD) do sistema armazena todos os (hiper)documentos. Cada documento possui uma parte de texto e outra de controle de links. A primeira destina-se ao editor, onde o texto é manipulado, enquanto que a segunda destina-se ao "engenho" do hipertexto, onde é feito o tratamento de links. O editor e o "engenho" do sistema estão interligados. Assim, a manipulação de uma referência pelo editor ativará o tratamento do link correspondente através da porta de comunicação.

Como estamos lidando operacionalmente apenas com o "engenho", a Base de Dados que definimos consta unicamente do controle sobre os links. Usando apenas esta parte de controle, especificamos os objetos:

Especificação dos Objetos

```
BD = ff (id_doc, texto)
documento = pair (id_doc, texto)
texto = pair (controle, subtxs)
id_doc = Atom      * identificador do documento
controle = ff (num_cria, contrl_ref)
contrl_ref = pair (num_loc, seq (ref))
             * controle de refer. de um parágrafo
num_cria = Nat
num_loc = Nat
ref = tuple (id_palav_chave, tipo_ref, id_doc_dest)
id_palav_chave = pair (palav, num_ocor)
          * idendificador da palavra de referência
id_doc_dest = Nat U pair  (id_doc, id_parag_dest)
          * identificação do documento destino
tipo_ref = 'intra U 'inter U 'subtx
num_ocor = Nat     * num. de ocorrência da palavra
                   de referência no parágrafo
id_parag_dest = Nat    * num. de criação do parag. dest
subtxs = ff ( num_cria_subtx, parag_subtx)
num_cria_subtx = Nat    * num. de criação do subtexto
parag_subtx = Atom    * parágrafo do subtexto
```

Espec. dos Objetos (cont.)

```
pilha_naveg = seq (naveg)
naveg = tuple (id_doc,contexto)
contexto = Atom    * contexto de edição
tabpag = ff (num_pag,paragrafos)
paragrafos = seq (num_parag)
num_pag = Nat    * número da página
num_parag = Nat    * numero do paragrafo
```

A Base de Dados é modelada por uma função finita (*ff*) que mapeia o identificador do documento (id_doc) com o texto correspondente. Foi definida como função porque esta correspondência é unívoca. Os hiperdocumentos são únicos e cada um está relacionado com seu controle.

O texto é um par (*pair*) que contém o controle dos links, parágrafo a parágrafo, e subtextos internos ao texto. O controle está descrito como uma função finita que faz a correspondência entre o num_cria e o num_loc juntamente com o controle de referências do parágrafo (contrl_ref).

Cada parágrafo é identificado pelo seu número de criação e de localização. O primeiro é único e imutável, criado por causa das referências. Após ter-se feito referência a um determinado parágrafo, poderá haver inserção e remoção de parágrafos anteriores a ele, o que provavelmente ocasionaria mudança em sua localização. Enquanto num_cria é invariante para o parágrafo, num_loc é localizador do parágrafo no texto.

O contrl_ref é formado pelo num_loc e pela seqüência (seq) de referências (ref) do parágrafo. A ref é uma tupla (*tuple*) formada por identificação da palavra chave (id_palav_chave), o tipo_ref, tipo da referência, e finalmente o identificador do documento destino (id_doc_dest). Este último é formado por um número natural (*Nat*) no caso das referências intra-documentos, ou uma tupla com o id_doc destino e o respectivo identificador de parágrafo. Os outros objetos são explicados em um relatório técnico mais extenso [8].

Para a manipulação dos links são necessárias operações realizadas sobre referências e parágrafos, as unidades básicas do sistema:

Modelo Abstrato das Operações

```
ConstrRef: tipo_ref x palav x id_doc_dest → ref
  * Constrói o registro de referência
InsRef: num_loc x tipo_ref x palav x id_doc_dest → documento
  * Insere o registro de referência no parágrafo correspondente
RemRef: num_loc x palav x parag → documento
  * Remove uma referência
CopiaRef: palav x num_loc x num_loc x parag x parag → documento
  * Copia uma referência
TransfRef: palav x num_loc x num_loc x parag x parag → documento
  * Transfere uma referência
InsParag: num_loc → documento
  * Insere um parágrafo
RemParag: num_loc → documento
  * Remove um parágrafo
TransfParag: num_loc x num_loc → documento
  * Transfere a localidade de um parágrafo
CopiaParag: num_loc x num_loc → documento
  * Copia um parágrafo
!EmpilhaNaveg: id_doc x contexto → pilha_naveg
  * Põe um registro de aportamento na pilha de navegação
PosicionaRef: num_loc x palav x parag x tabpag → num_pag
  * Posiciona a pagina da dada referencia
!PontoRetorno: → naveg
  * Recupera o ponto de retorno na navegação
```

No protótipo foram definidas funções para a criação da Base de Dados, criação e remoção de textos, etc, que interagem com a parte do sistema descrita neste artigo.

Para a criação de um link são usadas duas funções: ConstrRef e InsRef que fazem a construção e inserção da referência respectivamente. Especificamos estas funções como:

```
ConstrRef (tipo_ref,palav,parag,id_doc_dest) ≡
    if tipo_ref = 'inter
      then   ConstrRefInter (tipo_ref,palav,parag,
                  first(id_doc_dest),second(id_doc_dest))
      else   ConstrRefIntra (tipo_ref,palav,parag,
                                id_doc_dest)

InsRef (num_loc,tipo_ref,palav,parag,id_doc_dest) ≡
  let
       controle= Contrl_texto(!texto ())
    and reg_ref = ConstrRef (tipo_ref,palav,parag,id_doc_dest)
    and num_cria= ObtemNumCria (num_loc)
    and refs_parag = second (controle [num_cria])
    and seq_refs = AtualContr (refs_parag,reg_ref,1)
  in
    [nom_doc () →
     <controle ⊛ [num_cria → <num_loc,seq_refs ↑ reg_ref>],
                 Contrl_subtx (!texto ())>]
```

A função ConstrRef é definida por ConstrRefInter, caso a referência

seja inter-documentos, ou por ConstrRefIntra caso seja intra-documento. A InsRef insere uma referência no controle do parágrafo correspondente.

Usamos nas funções acima operadores presentes em *metoo:* o operador ⊕ faz a sobreposição de funções, let define objetos locais, first e second dão como resultado o primeiro e segundo elementos da tupla, respectivamente, [x → y] mapeia um elemento do domínio, x, com um do contradomínio, y, e ↑ concatena duas seqüências finitas. A função auxiliar ObtemNumCria retorna o num_cria do parágrafo a partir do num_loc, a Contrl_texto retorna o objeto controle e a Contrl_subtx, o subtxs a partir do objeto texto, obtido através da função !texto[2], e a AtualContr atualiza o número de ocorrência das outras palavras de referência, caso exista alguma palavra igual à de inserção.

Metoo possui uma notação abstrata e outra concreta, existindo uma tradução direta entre as duas. Assim, *metoo* abstrato pode ser transcrito para *metoo* concreto [8] de forma a ter um protótipo executável. Por exemplo, ConstrRef e InsRef são traduzidas como:

```
(defun ConstrRef (tipo_ref palav parag id_doc_dest)
    (if (eq tipo_ref 'inter)
        (ConstrRefInter tipo_ref palav parag (first id_doc_dest)
                                             (second id_doc_dest))
        (ConstrRefIntra tipo_ref palav parag id_doc_dest)))
(defun InsRef (num_loc tipo_ref palav parag id_doc_dest)
   (defobj documento
     (let
      (let
       (let
         (pair (nom_doc)
               (pair (plus controle
                     (makeff num_cria (pair num_loc
                                 (append seq_refs (list reg_ref)))))
                     (Contrl_Subtx (!texto)))
        (seq_refs.(AtualContr refs_parag reg_ref)))
       (refs_parag.(second (ap controle num_cria))))
      (num_cria.(ObtemNumCria num_loc))
      (reg_ref.(ConstrRef tipo_ref palav parag id_doc_dest))
      (controle.(Contrl_texto (!texto))))))
```

---

[2] o uso da exclamação (!) precedendo o nome da função, define que a mesma utiliza objetos pré-definidos que não são "passados" como parâmetro.

O operador (pair x y) constrói o par <x,y> de uma relação, (defobj x e) calcula a expressão e e armazena o resultado no objeto x (no caso do hipertexto, o documento - elemento de BD que está sendo manipulado), o operador ⊗ é transcrito como (plus f1 f2), ↑ como (append l1 l2), e [x → y] como (makeff x y).

A especificação de todas as outras funções do protótipo, incluindo a especificação em *meloo* concreto (protótipo executável já implementado) é descrita em [8].

7. CONCLUSÃO

O Sistema de Hipertexto em si é o "engenho" que gerencia os links, versões, segurança e visualização de hiperdocumentos. O usuário "trabalha" com uma *interface* específica para a sua área de interesse.

O nosso interesse particular é a concepção de um ambiente de desenvolvimento de *software* suportado por um Sistema de Hipertexto. Para isto, serão desenvolvidas *interfaces* específicas para manipular especificações e programas, bem como o "engenho" central do sistema.

A especificação funcional do gerenciador de links é o passo inicial para que possamos avaliar os elementos e as operações envolvidas. A partir desta especificação, será implementado o gerenciador de links da nossa versão inicial do Sistema de Hipertexto. Para isto, utilizaremos a linguagem SMALLTALK [13] devido a sua capacidade de modelar façilmente as trocas de mensagem existentes entre o "engenho" e as *interfaces* envolvidas.

8. REFERÊNCIAS

[1] Jeff Conklin, *A Survey of Hypertext*, MCC TR, Nr. STP-356-86, Rev 1, Austin, TX, Feb 1987.

[2] P. J. Brown, *Hypertext : The Way Forward*, University of Kent, Canterbury, 1987.

[3] L. Nancy Garrett, Karen E. Smith and Norman Meyrowitz, *Intermedia: Issues, Strategies and Tactics in the Design of a Hypermedia Document System*, IRIS, Brown University.

[4] Norman Delisle and Mayer Schwartz, *Neptune: A Hypertext System for CAD Applications*, ACM, 1986, pp 132-139.

[5] C. B. Jones, *Systematic Software Development Using the VDM Approach*, Prentice-Hall Intl.,1986.

[6] G. J. Steele, *An Overview of Common Lisp*, Proc. of the ACM Symp. on Lisp and Func. Prog., pp 98-107, 1982.

[7] P. Henderson, *meloo - A Language for Software Specification and Model-Building-Preliminary Report*, Univ. Stirling, Dec. 1984.

[8] A. M. L. Vasconcelos, A. C. V. Melo, E. S. Albuquerque e S. R. L. Meira, *Links em um Sistema de Hipertexto* - RT-DI-UFPE-013/88, Dep. de Informática, UFPE, 50739, Recife, PE, 1988.

[9] S. R. L. Meira, *Programação Funcional*, V JAI, VI Congresso da SBC, Recife-Olinda, PE, Jul 1986.

[10] S. R. L. Meira, *Engenharia de Software, Especificações Formais e Programação Funcional*, RT-DI-001/88, Dep. de Informática, UFPE, 50739, Recife, PE, 1988.

[11] F. M. Martins, *Especificação Formal e Prototipificação de Software* - A METODOLOGIA METOO, CCES-FM:R1/86, Univ. do Minho, Braga, Portugal, Mar 1986.

[12] I. Hayes, *Specification Case Studies*, Prentice Hall Intl., UK, 1987.

[13] A. Goldberg and D. Robson, *Smalltalk 80: The Language and its Implementation*, Addison Wesley, 1983.