Hipertexto: O Projeto do Sistema H

Silvio Meira, Judith Kelner, Eduardo Albuquerque, José Martins, Ana Melo e Alexandre Vasconcelos

Departamento de Informática
Universidade Federal de Pernambuco
CP 7851, 50739 Recife - PE - Brasil

Resumo

Este artigo apresenta alguns conceitos básicos de hipertexto, alguns dos principais progressos realizados na área, no mundo, até agora, e algumas idéias sobre o que poderá vir no futuro próximo.

Em particular, o texto é uma introdução ao trabalho sendo realizado no Grupo de Linguagens e Engenharia de Software (LinES) do Departamento de Informática da UFPE. O principal projeto em andamento na UFPE visa definir e implementar um sistema de hipertexto cuja fundamental aplicação deverá ser a gerência de processos de desenvolvimento de software. A curto prazo, um sistema de hipertexto voltado a aplicações de documentação eletrônica já está sendo implementado. Tal sistema é parcialmente descrito neste artigo.

1 Introdução

Um *hipertexto* é um documento organizado de forma não linear. Os mesmos conceitos de seqüência dos documentos em papel a que estamos acostumados, como capítulos, seções, páginas e parágrafos, também valem para *hipertextos*. Um *hipertexto*, no entanto, é um *documento eletrônico*, montado sobre uma estrutura não linear (um grafo) e pode ser usado de forma não trivial em relação aos documentos em papel.

Referências a outras partes do documento, ou a outros documentos, são possíveis através de *links*, ou *conexões*, que relacionam os *nós* de informação. Em primeiro lugar, estas conexões não precisam estar na ordem seqüencial do texto. As conexões podem ser puramente *sintáticas*, de forma que:

- a partir de um *button*, ou ponto de origem de uma referência, pressionado por um *mouse*, chegamos —via a abertura de uma janela que contém a página de hipertexto *apontada* pela conexão— a um novo nó.

ou semânticas:

- um *nó* contendo uma *especificação* de um sistema pode estar relacionado a outro que contém a *implementação* do mesmo. Alterações em cada um (detectadas pelo sistema de hipertexto) implicariam em modificações no outro, administradas pelo sistema.

Estes são apenas dois dos possíveis tipos de relações, e pode haver uma infinidade, tanto quanto os diferentes tipos de informação que podem ser armazenadas em tais sistemas.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

1.1 Conceitos Básicos

A maioria dos sistemas atuais de edição eletrônica de texto trata documentos similares aos livros comuns, isto é, com palavras armazenadas de forma linear e não relacionadas entre si.

Na realidade, isto é muito menos do que se espera quando se tem o poder computacional das estações de trabalho modernas à disposição. Na maioria das vezes, quando estamos lendo um determinado "assunto" em um livro com características de 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 [16].

A necessidade de armazenar e recuperar textos com estrutura não linear fez surgir um novo conceito de armazenamento de informação suportado por computador, onde documentos que têm referências a outros documentos ou a trechos não seqüenciais do próprio documento (notas de "rodapé", referências a seções ou figuras do texto, etc.) são manipulados (editados e/ou visualizados) numa tela de alta resolução. Sistemas com tais características são denominados *Sistemas de Hipertexto*, e documentos editados e armazenados nestes ambientes, *hiperdocumentos* [12], [6], [7], [8], [9], [16] ou *hipertextos*.

A essência dos sistemas de hipertexto está na sua capacidade de armazenar e recuperar informações não lineares de uma forma eficiente. A idéia básica de hipertexto pode ser expandida de modo que os elementos interligados possam ser não somente textos, mas também imagens, sons, programas fontes e código executável.

Nos Sistemas de Hipertexto as ligações inter ou intra documentos são estabelecidas por *links*, que tem a finalidade de endereçar objetos chamados "nós", que são as unidades de referência básica em um documento. Os nós são também as unidades elementares de armazenamento de texto, imagem, animação, etc.

1.2 Ações em Sistemas de Hipertexto

Para dar uma melhor idéia do comportamento de um sistema de hipertexto, discutiremos algumas operações possíveis. Alguns sistemas têm mais recursos que os mostrados, outros menos, ou diferentes, e a discussão aqui não é exaustiva, servindo apenas como exemplo curto.

Vamos assumir que um sistema de hipertexto genérico é capaz de administrar bibliotecas de documentos, mais ou menos no estilo de um banco de dados relacional. Daí, concentrarmos nossa atenção na manipulação de nós e links pelo sistema, o que vem a ser justamente a diferença entre editores "normais" e sistemas de hipertexto.

Os nós de um hiperdocumento podem ser criados, removidos, alterados, copiados, transferidos ou editados. O sistema de hipertexto deve garantir a integridade dos documentos, nós, links e mapas nos casos de remoção, alteração, cópia e transferência de nós e links.

Um nó pode ser composto por "objetos complexos", como código executável. Neste caso, por exemplo, uma referência ao mesmo ativará o código, via um comando do tipo *load*. Isto funciona exatamente como abrir uma janela para o sistema operacional, mas as possibilidades são muito mais interessantes se consideramos, por exemplo, a existência de links dentro de código objeto.

O acesso a um nó pode ser feito diretamente através de um button, que tanto pode ser texto como qualquer outro tipo de *região* sensível ao toque de um mouse, por exemplo. Outra forma de se chegar a um nó é através de um comando de busca de palavras chaves previamente estabelecidas na criação do documento —o que é bastante eficiente— ou através de busca de strings, como em editores de texto, dentro do documento.

Considerando que hiperdocumentos podem ter, em geral, estrutura muito complexa, especialmente se estão relacionados com outros documentos, é fundamental que se disponha de um

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

mecanismo para se *navegar* na estrutura, abrindo assim a possibilidade de acesso direto a qualquer nó visível em um *mapa* do texto. Este mapa equivale a uma carta geográfica, onde as cidades são os nós e as estradas os links entre eles. O *navegador* ou *graphic browser* também pode ser usado para criar, durante a execução, uma trilha superposta ao (ou parte do) mapa, mostrando onde o usuário está situado no momento, dando o *contexto* e a história da leitura atual.

Os nós podem ter *rodapés*, que são parte integral dos mesmos, mas se mantém invisíveis até o momento de sua ativação. Os rodapés não têm status de nó e desaparecem quando o *click* que os ativou deixa de existir. Também pode haver, em um nó, textos chamados *substituições*, que tomam, no texto, o lugar antes ocupado por seus buttons, quando estes são ativados.

A base de dados que suporta o hipertexto é um grafo de nós de texto (ou imagem, animação, sons...). As janelas mostradas na tela podem possuir uma correspondência um a um com nós na base de dados mas apenas alguns nós estão abertos como janelas a qualquer momento. As janelas podem ser reposicionadas, fechadas, ter seu tamanho alterado ou "postas de lado" (temporariamente abandonadas), caso em que passam a ser representadas por ícones (imagem simbólica correspondente à janela). A posição, tamanho, forma e cor da janela ou do ícone podem servir para lembrar seu conteúdo ou tipo. O fechamento de uma janela faz com que as alterações feitas sobre a mesma sejam salvas antes que ela desapareça. Ao se ativar o ícone de uma janela, a mesma é normalmente aberta, *instantaneamente*, na posição que estava antes de ser abandonada.

O usuário pode criar novos links para novos nós (anotações, comentários, etc.) ou para *nós* existentes, estabelecendo novas conexões. Em ambientes onde um conjunto de pessoas *coopera* na criação de produtos, esta capacidade de relacionamento é fundamental para o bom andamento do trabalho. Espera-se que este seja um dos grandes usos de sistemas de hipertexto no futuro, substituindo os papéis e as formas ultrapassadas de correio eletrônico hoje em voga.

2 Hipertexto na Prática

A concepção prática de sistemas de hipertexto é relativamente recente, porém a idéia original datada de 1945, quando Vannevar Bush concebeu *Memex*[16], como um suplemento para a memória do usuário humano, onde textos e gráficos (notas, fotografias, desenhos, etc.) seriam armazenados em um esquema de índices com a mesma funcionalidade do que hoje se chama links de hipertexto.

Um fato importante é que apesar de não dispor do computador digital na época, Bush conseguiu os efeitos desejados usando microfilmes e fotocélulas.

Segundo Jeff Conklin em [16], os sistemas de hipertexto podem ser classificados em quatro grandes grupos:

- Sistemas Macro-literários: tecnologias que suportam bibliotecas *on-line* em grande escala, onde ligações entre documentos são suportadas por máquinas.
- Ferramentas para exploração de problemas: recursos que auxiliam a estruturação e convergência de idéias na solução de problemas.
- *Browsing Systems*: semelhante aos sistemas literários, mas em menor escala.
- Sistemas de hipertexto genéricos: este tipo de sistema possui propósitos genéricos e, pelo menos em teoria, suportam os mais diversos tipos de aplicação.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

Sistemas Macro-Literários

Os sistemas literários em larga escala buscam a integração dos vastos volumes da literatura existente, de forma dinâmica. Aqui se inclui, por exemplo, o pioneiro *Memex*.

Dentro deste mesmo grupo de classificação, duas décadas após a proposta de Bush, Douglas Engelbart projetou o NLS (On-Line System)[16] influenciado pelas idéias de Bush, para o *Stanford Research Institute*. Este sistema tinha como objetivo principal o uso de computadores como um novo estágio na evolução do intelecto dos seres humanos.

O NLS, cuja versão atual é chamada *Augment*, tem a capacidade de armazenar documentos e, tal como uma rede de comunicação, compartilhar um espaço de trabalho para o planejamento e desenvolvimento de projetos. Foi desenvolvido em um DEC-20, e armazena informações em uma sofisticada estrutura hierárquica que permite desvios não hierárquicos. Para o acesso "imediato" às informações, Engelbart foi o pioneiro na adoção do *mouse* como dispositivo de entrada, no caso usado para navegação na rede. O NLS tem como base três aspectos principais:

- uma Base de Dados (não linear) para armazenamento de informação,
- um conjunto de *filtros* para recuperação de informação a partir da Base de Dados, e
- visões (*views*) que estruturam a informação a ser exibida em uma tela de alta resolução, onde cada janela possui exatamente um nó.

Na realidade, este foi o primeiro sistema computadorizado de informação integrada com múltiplas janelas e tela compartilhada, e introduziu a noção de editores estruturados em um ambiente distribuído.

O termo *hypertext* foi adotado por Ted Nelson na década de 60, quando fez a definição do sistema *Xanadu*[16]. A idéia central era a criação um ambiente literário em escala global. Este talvez seja, dos sistemas iniciais, o sistema de hipertexto mais conhecido e foi utilizado por um grande número de pessoas para criar, interagir e interconectar textos. Uma característica marcante deste sistema é a separação que existe entre a *interface* do usuário e a base de dados do sistema, à qual é dada maior ênfase.

Um outro sistema importante dentro deste mesmo grupo de classificação é o TEXTNET [10], desenvolvido por Trigg, na University of Maryland. Tem como objetivo principal dar suporte a textos escritos em forma não linear, ou seja, cada documento está organizado em partes mais primitivas de texto, interligadas através de links (similar a uma rede semântica). Este sistema implementa dois tipos básicos de nós: os *chunks* que são nós textuais (possuem texto linear), e os *tocs* que são nós organizados hierarquicamente. O acesso é feito através da navegação de links, ou por meio da junção dos nós textuais de modo a formar um texto linear. Uma outra característica importante neste sistema são os *paths*, ou caminhos pré-estabelecidos para a navegação. Um mesmo documento poderá ter mais de um caminho associado, organizados mediante os níveis de interesse do usuário, ou seja, o usuário poderá ter várias "visões" de um mesmo documento.

Além destes sistemas, existem alguns outros que possuem os mesmos propósitos, tal como o *Guide*[15], desenvolvido na Universidade de Kent, hoje disponível em Macs e PCs.

Sistemas Exploratórios

Outro tipo de sistema existente é o que dá suporte à exploração de problemas e soluções. Uma importante característica da maioria destas ferramentas é a facilidade de suprimir detalhes em vários níveis, possivelmente especificados pelo usuário.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

O IBIS (Issue Based Information System) [13], idealizado por Horst Rittel, introduz a idéia de sistemas utilizados para a análise de problemas "danados"[1]. Segundo Rittel, estes são problemas que não podem ser resolvidos por métodos de análise tradicionais, ou seja, não se consegue resolver seus elementos em separado e o "caminho" é resolvê-los por completo, em um ataque do tipo "força-bruta organizada".

Soluções para este tipo de problema envolvem muitas trocas de idéias e questionamentos sobre as mesmas, pontos de vista, valores e propósitos. Este procedimento propicia um entendimento comum para as mais importantes questões e suas implicações emergentes.

IBIS possui três tipos de nós (*issues*, *positions* e *arguments*) e nove tipos de links (semânticos) para interconectar estes nós: *responds-to*, *questions*, *supports*, *contradicts*, *objects-to*, *specializes*, *generalizes*, *refers-to* e *replaces*. Existe, ainda, uma classificação para os links em relação à criação. Links primários são os que conectam, automaticamente, o nó à rede do sistema, enquanto que links secundários são os estabelecidos pelos usuários.

Um outro sistema com estes mesmos objetivos foi desenvolvido na Universidade da Carolina do Norte (UNC), o WE [16], baseado no modelo cognitivo de comunicação de processos. WE é estruturado internamente por uma rede de idéias e externamente, a princípio, é organizado em estrutura hierárquica, a qual é "decodificada" para uma estrutura de fluxo linear. O sistema possui uma visão gráfica e outra hierárquica, e sua rede de links e nós está armazenada em uma base de dados relacional.

Tem-se ainda nesta mesma categoria os *outline processors*, o SYNVIEW, desenvolvido por David Lowe[16], etc.

"Folheadores"

Uma outra categoria de sistemas de hipertexto, dentro da classificação de Conklin, são os direcionados ao acesso rápido à informação, para aplicações que envolvem uma grande quantidade de informação, os folheadores (*Browsing Systems*). Dentre estes está o ZOG/KMS [14], o *Hyperties*[16] e o Examinador de Documentos Simbólicos[16].

ZOG foi desenvolvido a partir de 1972 em Carnegie-Mellon, para o porta-aviões USS Carl Vinson. Consiste potencialmente de uma grande Base de Dados de pequenos segmentos vistos ao mesmo tempo. Foi desenvolvido com o objetivo de servir simultaneamente a um grande número de usuários e projetado para operar em terminais padrão num sistema de *Timesharing*, e é usado como um centro de documentação (e operação) do porta-aviões.

A partir de 1981 este sistema foi reestudado, o que gerou o KMS (Knowledge Management System). Aqui, cada segmento do sistema é denominado *frame*, o qual é rotulado e cada tela associada possui uma série de comandos (por exemplo: *edit*, *help*, *back*, *next*, *mark*, *return*, etc). A estrutura é geralmente hierárquica, podendo conter links com referências cruzadas. Além disso, um item pode conter um *frame* relativo a ativação de um processo. A grande utilidade deste tipo de sistema é que o usuário dificilmente se desorienta dentro de seu contexto desejado, já que o movimento entre os *frames* é muito eficiente, e ele tem a possibilidade de se reorientar mediante pouco esforço, no caso de algum erro.

Sistemas de Propósito Geral

A quarta e última classe descrita é a de sistemas de hipertexto de propósito geral, que possuem uma ou mais linhas de ação. Seu objetivo é o experimento do próprio conceito de hipertexto

[1] *Wicked Problems.*

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

como tecnologia. Como exemplo destes, tem-se Neptune [9], Intermedia [12], NoteCards [11], Boxer [16], CREF [16], entre outros.

Um dos importantes sistemas deste grupo é Intermedia da Brown University. Intermedia é o resultado de duas décadas de trabalho, com três gerações do sistema. O primeiro sistema, "Hypertext Editing System" data de 1968; segundo foi FRESS (File Retrieval and Editing System), que tem como característica links com referências bidirecionais e dinamicamente hierárquicos; e, por fim, o "Eletronic Document System", que enfatiza a parte gráfica e de auxílio à navegação.

Intermedia dá ao usuário a possibilidade de criar links sofisticados entre os documentos, e tem como objetivo, o auxílio à educação, através de um software avançado, e melhoramento da base metodológica e tecnológica na qual o software é construido. Tecnicamente, o sistema tem como objetivo a extensibilidade, reusabilidade, consistência, modularidade e facilidade de entendimento. Tem estrutura hierárquica de documentos e, como características marcantes, a idéia de "contexto" e seleção de links através de seus atributos.

Intermedia encoraja o trabalho colaborativo pela facilidade que oferece aos usuários para examinar e editar documentos compartilhados entre projetos. O controle de acesso assegura a integeridade destas informações e permite que os indivíduos trabalhem juntos para atingir um objetivo. Oferece também a possibilidade de desfazer as operações executadas desde o início de uma sessão.

3 Hipertexto em Desenvolvimento de Software

Apesar de podermos eventualmente classificar os sistemas deste tipo como sistemas de propósito geral, as peculiaridades do processo de desenvolvimento de software são tais que vale a pena abrir um espaço para os sistemas de hipertexto que suportam desenvolvimento de software.

Como é sabido, o custo do Software suplanta largamente o custo de Hardware em um sistema de computação. Entre os vários fatores que levam a isto, um dos mais importantes é a complexidade dos documentos envolvidos e suas possíveis interações. Aqui falamos não só dos documentos literais, mas de forma generalizada, considerando especificações informais e formais, trechos de código fonte e objeto, suas documentações, versões, visões, etc. Devemos também levar em consideração que nem toda informação necessária para a documentação de um Sistema de Software pode ser embutida no código do sistema. Quanto maior o sistema de software mais críticos são os problemas de consistência de documentos, completude, controle de revisões e recuperação eficiente.

De uma certa forma, mesmo que resolvamos boa parte dos problemas de desenvolvimento rigoroso de software, do ponto de vista teórico, há de haver meios efetivos de integrar as partes do processo de desenvolvimento, manutenção e possivelmente uso.

Ainda mais, no caso particular dos sistemas de software, justamente por serem considerados objetos "maleáveis" pelos seus usuários, os sistemas, particularmente os de uso comercial, são muito dinâmicos, tendo que se adaptar rapidamente às mudanças legais, econômicas, organizacionais, etc.

Outro problema comum que leva ao aumento do custo de software é a tendência à "reinvenção da roda". Uma série de textos, formais ou não, que são comuns a várias partes do projeto, ou mesmo a projetos distintos, tendem a ser reescritos cada vez que são utilizados e, a cada alteração de um deles, há necessidade de se repetir a alteração em pontos diferentes. Na prática, o fato é que isto nunca é feito como deveria, desintegrando a grande maioria dos projetos reais.

Vejamos alguns exemplos de problemas onde poderíamos aplicar sistemas de hipertexto.

Atualmente, o software atingiu tal grau de sofisticação e complexidade que seus manuais são, normalmente, divididos em alguns volumes, com referências cruzadas a cada página. Além

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDF

disso, é comum desenvolver software em diversas versões, para diversos tipos de máquinas e/ou usuários. A cada uma dessas versões está, possivelmente, associada uma versão diferente do manual. Aqui poderíamos usar hipertexto para documentação integrando os manuais em uma biblioteca eletrônica.

Olhando para os programas, um mesmo código fonte pode produzir mais de um executável. O mesmo código básico de um compilador, por exemplo, dependendo da presença ou não de determinados trechos, produz compiladores com maiores ou menores recursos. Desta maneira, cada usuário, dependendo de sua "visão", poderá trabalhar com um compilador diferente. Aqui, todo o controle de *visões*, *versões* e *relações* com a documentação pode ser codificado em um hipertexto.

Da mesma forma, uma equipe de desenvolvimento pode operar com partes independentes de um mesmo projeto, usando o mesmo documento fonte, mas cada membro tendo acesso à visão e versão específicas na qual está trabalhando. Este seria o uso "normal" de sistemas de hipertexto como administradores de bibliotecas eletrônicas.

Isoladamente, cada um destes usos é de grande auxílio ao processo de desenvolvimento, mas não se comparam ao que pode ser feito quando se usa sistemas de hipertexto para *integrar* todo o ambiente de desenvolvimento de software.

Um grande esforço de pesquisa vem sendo dispendido em ferramentas de desenvolvimento de software. Assim é que vimos surgir editores orientados por sintaxe, editores gráficos para diagramas que fazem parte das fases do processo de desenvolvimento de software, sistemas de controle de versões e muitas outras ferramentas destinadas aos projetistas de software. Porém, quase todas estas ferramentas se caracterizam pela total independência uma da outra.

Um Sistema de Hipertexto pode ser uma ferramenta muito útil em Ambientes para Desenvolvimento de Software, sendo usado para integrar as partes do ambiente, proporcionando uma interface única e amigável, que ao mesmo tempo facilite o desenvolvimento e documentação .

O trunfo principal de sistemas de hipertexto como auxílio ao desenvolvimento de software é a possibilidade de interligar especificação, implementação e documentação de todo o sistema. Deste modo, pode-se ter especificações conectadas a programas correspondentes e vice-versa, e ainda, ligadas à respectiva documentação. Este tipo de organização facilita o entendimento do sistema, permite o acesso "imediato" à informação e possibilita um conhecimento em vários níveis, do mais abstrato (abrangente) ao mais detalhado.

Por exemplo, em um sistema real, os programas podem estar em nós de hipertexto que estão ligados ao editor orientado à sintaxe apropriado. Assim, um programador ao editar um programa automaticamente tem acesso ao editor apropriado da mesma forma que o documentador (se ainda existir) ao alterar um diagrama ou um relatório usará a ferramenta adequada.

Um outro exemplo, no caso de alteração da definição de uma parte do sistema, o analista pode ter uma idéia do impacto sobre o sistema como um todo ao fazer com que o sistema de hipertexto monte graficamento todos os ponteiros *buttons* que saem do nó alterado. Se desejar maiores detalhes, pode navegar pelos nós que são mostrados. Desta forma, terá em pouco tempo uma idéia quase perfeita sobre as alterações nescessárias tanto nos programas como na documentação. Uma vez decidida a alteração, o hipertexto ainda informará aos programadores e documentadores quais partes de seus trabalhos foram afetadas agilizando e facititando o processo de alteração.

Um outro ponto relevante no uso de sistemas de hipertexto como suporte a ambientes de desenvolvimento de software, é a possibilidade de manter restrições de integridade entre programas, especificações e documentação de sistemas através de um processo semi-automático.

Estando as partes do sistema interligadas, a cada modificação em um objeto que depende de outros, poderão ser ativados processos mantenedores de integridade, de modo que a informação

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

correspondente seja devidamente modificada, de forma automática ou não.

3.1 System Factory

Um excelente exemplo do uso de sistemas de hipertexto é dado pelo *System Factory* [4] da University of Southern California, ao qual estava associado Pankaj Garg.

Um dos problemas que existe na documentação de um software de larga escala é o de determinar o que deve ser documentado. O método de documentação do System Factory tenta resolver este problema advogando uma mistura de preocupações técnicas e organizacionais.

System factory é um método de documentação que divide o processo de desenvolvimento do software em atividades, e cada uma destas possui a documentação associada. A *documentação final* de cada projeto possui oito documentos, os quais correspondem a cada uma das atividades pré-estabelecidas: Especificação de Requisitos, Especificação Funcional, Especificação da Arquitetura de Projeto, Especificação de Detalhes de Projeto, Documento do Código Fonte, Documento de Teste e Qualificação, Manual do Usuário e Manual de Manutenção do Sistema.

A fase de Especificação de Requisitos descreve os requisitos operacionais e os não operacionais do projeto. Os operacionais apresentam o desempenho através do projeto e implementação do sistema; enquanto que os não operacionais definem seus recursos organizacionais. A fase de Especificação Funcional define as funções computacionais do sistema, desenvolvendo-as de forma incremental: especificação informal, forma gráfica do sistema e ambiente e, por fim, a especificação formal na linguagem de especificação GIST[4].

As fases seguintes são: Especificação da Arquitetura que descreve a interconexão da estrutura dos módulos do sistema; a Especificação detalhada do Projeto, que descreve o comportamento dos algoritmos, recursos e módulos do sistema definidos na fase anterior; o Documento de Código Fonte reflete, através deste código, a estrutura já descrita em documentos anteriores; Documento de Testes e Garantia de Qualidade armazena os resultados obtidos nos testes que comprovam a qualidade do sistema; e por fim, existe o Manual de Manutenção do Sistema

Estes elementos (documentos) do *System Factory* são suportados pelo sistema de hipertexto DIF (Documents Integration Facility [4]) que, além de manter a interface com o usuário, é responsável pelas ligações estruturais do *System Factory*, bem como pelas ligações estabelecidas pelo usuário.

DIF além de elaborar automaticamente a estrutura mais externa do projeto, descrita no método System Factory, proporciona a facilidade de integração e comunicação entre os elementos descritos. Possui como elemento fundamental de integração *links* e também algumas ferramentas que facilitam o desenvolvimento do sistema, como por exemplo, um editor orientado à sintaxe de GIST, um editor para NuMil[4] que é uma linguagem de definição de módulos e suas dependências (relações).

A existência de uma estrutura bem determinada e integrada do ciclo de vida do software neste sistema, facilita a produção, organização, armazenamento para pesquisa futura, reutilização e revisão de projetos.

4 O Sistema de Hipertexto H

O sistema H faz parte de um projeto do grupo de Linguagens e Engenharia de Software (LinES) da UFPE dentro de um estudo de manipulação de hiperdocumentos, ora em andamento.

O estágio inicial do projeto H é a implementação de um protótipo de um Sistema de Hipertexto com características sofisticadas, sem preocupação com performance. Inicialmente, H manipulará exclusivamente figuras (*graphics*) e textos. H está sendo implementado em uma lin-

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

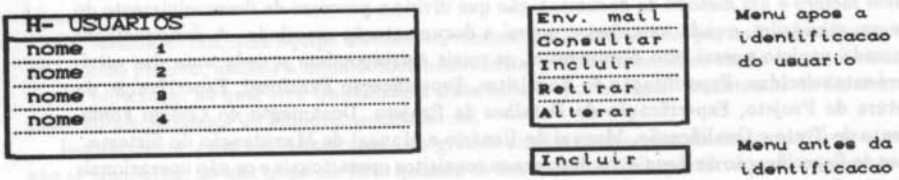

Figura 1: Tela Principal e Menu Usuário de H.

guagem orientada a objetos, Smalltalk, o que também está servindo para estudar a possibilidade de tentar uma implementação do sistema final em linguagens orientadas a objetos.

Com a experiência adquirida no desenvolvimento de H, pretendemos a médio prazo ter um novo protótipo (Hex [1]), mais sofisticado, derivado de uma especificação formal usando a linguagem Zc [2].

Este outro sistema, atualmente em fase inicial de modelo matemático, será o integrador das partes do ambiente para desenvolvimento de software ADRIS —Ambiente de Desenvolvimento RIgoroso de Software[3], que conta ainda com uma linguagem funcional (A [3]) para a geração de protótipos e uma metodologia de definição formal orientada a modelos baseada na linguagem Zc.

Este ambiente teria o sistema de hipertexto como base para sua integração e apoio ao desenvolvimento, tanto do ponto de vista da integração dos documentos associados ao processo de desenvolvimento de software, como da automatização do processo propriamente dito, de forma semelhante ao Ishys[5].

4.1 Características Gerais do Sistema H

A seguir, descrevemos as características gerais de H, algumas das quais já consolidadas e implementadas. Esta Seção deve ser lida mais como uma carta informal de intenções do projeto, do ponto de vista comportamental, e não como a definição final de H. Em alguns lugares, a descrição é bem mais operacional do que é necessário, o que é intencional. Se este fosse um hiper-artigo, só precisaríamos mandar uma hiper-carta (cf. Sec. 4.1.10) para os leitores!

Ao ser ativado, H mostra uma tela de apresentação com o símbolo H e uma janela chamada **H-Usuarios**, onde o usuário deve se identificar. O sistema lida com um ambiente multi-usuário, onde os objetos podem ser protegidos *à la Unix*.

A identificação pode ser feita via menu, com o usuário selecionando seu nome caso exista, ou incluindo um nome, caso tenha poderes para tal. Após a identificação, quando positiva, é ativada a janela **H-Documentos**, de onde podem ser realizadas operações sobre documentos.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompress

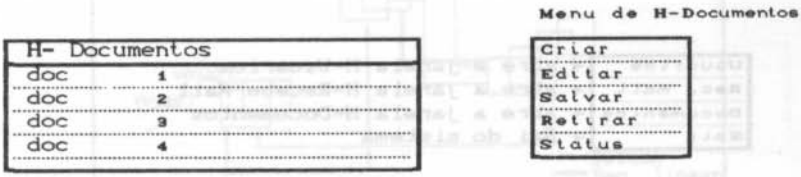

Figura 2: Janela H-Documentos.

No caso mais comum, edição de documento, é feita a seleção de um documento e em seguida deve-se escolher a função Editar do menu da janela.

O estado do sistema é dado pela n-upla

```
< correio,
  H-docs,
  usuarios >
```

Em `correio` é mantida uma forma de *correio eletrônico*. Além de cartas eletrônicas normais, é possível a um usuário enviar e receber hiper-cartas, que são nada mais nada menos que (partes de) hiperdocumentos.

`H-docs` contém os documentos (no formato `H`) que são reconhecidos pelo sistema. Um documento só poderá ser utilizado se estiver nesta relação. Nem todos documentos têm que ser completamente editados pelo `H`. É possível *importar* documentos escritos em editores de texto comuns e superpor a informação de estruturação e interligação de um hipertexto. Em particular, está-se estudando o problema de "compilação" de hiperdocumentos em um projeto separado[17], usando uma linguagem de Markup (LinDA).

em `usuarios` temos a identificação de todos os usuários cadastrados.

4.1.1 Menu Global

Este menu contém operações globais ao sistema e que podem ser utilizadas a qualquer momento, de qualquer lugar (janela) pelo usuário.

Vale ressaltar que os menus são ativados através de um *click* do mouse na janela correspondente, usando o estilo de Smalltalk.

Para as opções Consulta e Envia Carta, o usuário deve escolher um nome em **H-Usuários**. Esta pessoa é quem terá seus dados pessoais consultados ou será a destinatária da Carta.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

```
Usuarios    → abre a janela H-Usuarios
Rec. mail   → abre a janela H-Recebe Mail
Documentos  → abre a janela H-Documentos
Sair        → Sai do sistema
```

Figura 3: Menu Global.

4.1.2 H-Documento

Um *H-Documento* é informalmente modelado pela n-upla abaixo:

```
< ident-usuario,      // numero gerado pelo sistema para a identificacao
                         dos usuarios
  ident-documento,    // numero gerado pelo sistema para identificacao
                         do documento
  versao,             // contem as informacoes para identificar o
                         autor de cada versao do documento
  keywords,           // conjunto de palavras definidas como chaves para
                         acesso usando o comando ACHE
  conj-de-visoes,     // conjunto que identifica as diversas visoes
                         que o leitor pode ter do documento
  ultimo-no-criado,   // numero que identifica o numero do ultimo no criado
                         no documento. E de controle do sistema
  primeiro-no >       // ponteiro para o primeiro no do documento
```

A versão é uma lista onde cada elemento contém as seguintes informações :

```
< nro. versao,        // numero sequencial que identifica a versao do
                         documento
  id. do autor,
  data          >
```

Um documento pode ser aberto de duas formas: explicitamente, quando o usuário escolhe a opção **Editar** no menu 3, ou quando existe uma ligação (*link*) entre documentos. O documento contém um ponteiro para seu primeiro nó que será automaticamente aberto quando da abertura do documento. A partir daí cada nó *sabe* qual é seu próximo nó.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompress

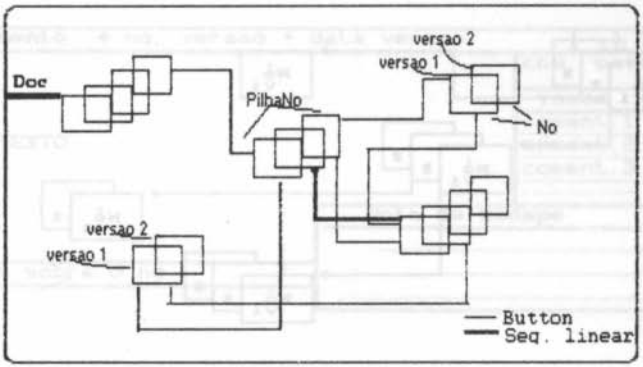

Figura 4: Exemplo de um H-documento.

4.1.3 Nós

Os nós contém *bit maps* que podem representar textos ou imagens (*graphics*). Um nó não pode conter textos e imagens ao mesmo tempo, isto porque a cada nó está associado um "processador". No entanto, há uma forma limitada de texto nos nós de imagem.

Um documento pode ser visto como uma estrutura linear com possíveis desvios ativados pelos *buttons*[2], que indicam ligações entre (partes do) texto sendo lido e outros trechos do mesmo texto ou de outro.

Um nó contém as seguintes informações:

```
< tipo,          // tipo do no (texto/graphics)
  id-documento,  // a que doc o no pertence
  id-no,         // numero do no
  conteudo,      // contem a pilha com as versoes do no
  data-cria   >  // quando foi criado
```

Conteúdo é uma *pilha transparente* com as seguintes informações:

```
< versao,        // numero da versao do no
  links-saindo,  // conjunto de links que saem do no
  id-visao,      // qual e o button de visao neste no, se houver
  prox-nos,      // conjunto dos proximos nos. Caso haja um button de
                 // visao, pode haver mais de um proximo no
  decisao,       // conjunto de buttons de decisao do no
  conteudo,      // texto/graphics
  comentario,    // comentario associado ao no
  rodape      >  // as "footnotes" para este no
```

O topo da pilha contém a versão mais atual do nó. A pilha é *transparente* porque o usuário pode ter acesso a diferentes versões do documento, sem ter que administrá-las diretamente.

[2] Um *button* é uma região da tela sensível ao mouse.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

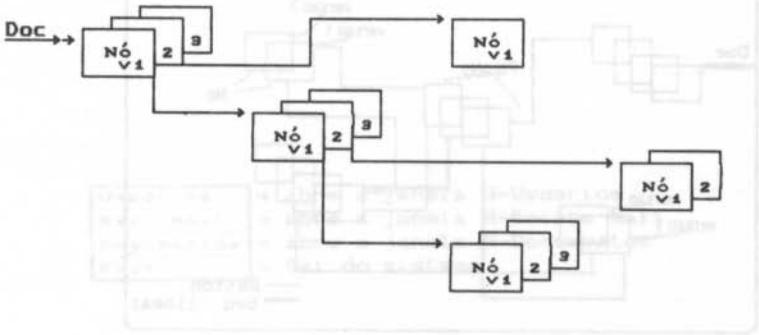

Figura 5: Versões de Documentos e Nós.

A remoção de um nó implica na criação de uma nova versão do nó com informação nula. Isto é feito para manter o controle sobre a seqüência do documento, nas diversas visões existentes. Adicionalmente, este tipo de remoção faz com que seja trivial a recuperação de nós removidos acidentalmente.

Há operações para inserção de nós com opção para *antes* e *depois* de um dado nó. Quando um nó estiver sendo editado, será permitido o acesso a outras versões do mesmo. No entanto, o sistema garante que o usuário não pode criar *duas novas versões* de um nó ao mesmo tempo.

O usuário tem a opção de descartar versões anteriores do nó. Caso escolha manter apenas a versão mais atual, apenas o topo da pilha será mantido e a versão do documento no nó será a de número 1.

O sistema mantém uma tabela com todos os nós que foram alterados para permitir que o usuário descarte as alterações que desejar.

4.1.4 Rodapés

As notas de rodapé (*footnotes*) fazem parte do nó. Não podem ter buttons ou ser destino de *links*. Quando for criada uma nova versão do nó, as suas notas de rodapé serão copiadas para a nova versão.

4.1.5 Montagem de uma Janela de Texto

Os buttons de referência devem ser verificados quando da montagem da janela. Para cada visão do documento, somente serão mostrados aqueles que são visíveis na mesma e não são nulos, isto é caso durante a operação algum button de visão seja alterado, o sistema remonta as telas quando o usuário as reativar.

A pré-definição de *buttons de visão* será possível a qualquer tempo durante a sessão. Caso o usuário não queira ter buttons válidos para toda sessão, pode fazer as escolhas quando estiver percorrendo o documento, o que permite que ele percorra um documento tendo sua visão particular.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

Figura 6: Rodapés em H-Documentos.

4.1.6 Comentários

A idéia de comentários em um sistema de hipertexto é possibilitar que "leitores" de um documento deixem comentários nas páginas do mesmo, tanto para o "escritor" como para outros leitores.

Os *buttons* correspondentes levam a *comentários ou explicações* que funcionam como anotações na margem de um livro. Ou seja, uma pessoa lendo um documento pode deixar *marcas* que levem a seus comentários sobre o texto. O usuário que está lendo o documento pode selecionar os comentários que interessam, fornecendo a identificação de seus autores. Neste caso, todos os *buttons de comentário* que tenham sido introduzidos por outros usuários se tornariam invisíveis.

O button de comentário faz parte do nó, e tem as seguintes informações:

```
< Id. Autor,
  data do comentario,
  texto >
```

Cada nó tem uma *seqüência* de comentários. Cada objeto acima descrito é um elemento desta seqüência. Quando for criada uma nova versão do documento, o usuário tem a opção de copiar ou não os comentários.

Existe a opção de *filtrar* os comentários, deixando visíveis apenas os comentários feitos por determinadas pessoas. Quando esta opção for escolhida , o usuário deverá fornecer uma lista com a identificação dos autores dos comentários. Os comentários podem ser mostrados classificados por nome do autor ou por data de criação. Ao escolher um dos comentários, será aberta uma janela do tipo Editor de Texto, com o texto.

4.1.7 Versões

Um dos pontos fortes de H é sua capacidade de manter um número arbitrário de versões de um documento. Estas versões são mantidas por nós do documento que é a unidade mínima de alteração em H. Os nós que formam os documentos de H são mantidos em uma estrutura do tipo

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

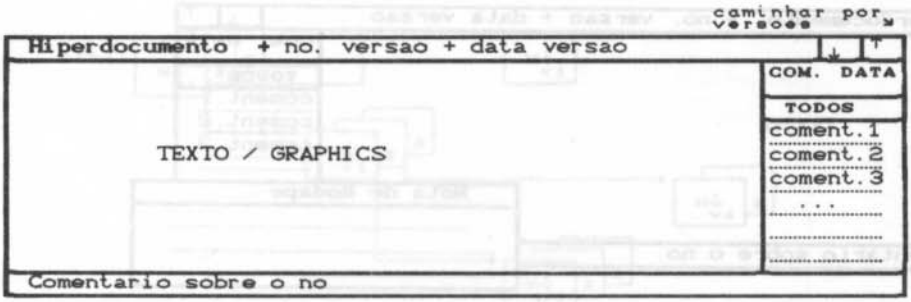

Figura 7: Comentários em H-Documentos

pilha transparente (uma instância da classe PilhaNo) onde cada elemento contém uma versão do nó (cada elemento é uma instância da subclasse de No apropriada). Todos os elementos da pilha devem ser do mesmo tipo, ou seja, um nó uma vez definido como Texto, por exemplo, deve conter texto em todas suas versões. A criação de uma nova versão se faz no momento que o usuário edita uma das versões já existentes do nó. Todas as informações existentes naquela versão são copiadas para o topo da pilha, criando uma nova versão do nó que possuirá o número da versão do documento mais um. Até que seja escolhida a opção SALVAR VERSAO no menu da janela H-Documentos, cada vez que o usuário editar aquele nó, estará trabalhando com esta versão que foi criada. Portanto, uma vez criada uma versão de um nó após um documento ter sido salvo, H não permite que se faça edições (exceto cópia de trechos) em outras versões deste nó pois isto faria com que existissem "duas versões atuais" diferentes o que provocaria uma inconsistência.

Não existe uma remoção física de um nó em H. Ao se fazer a remoção de um nó o que o sistema realmente faz é criar uma nova versão do nó com conteúdo nulo. Assim, dentro de um documento, este nó passa a funcionar (nesta versão) simplesmente como um redirecionador dos apontadores que nele chegam para o próximo nó na seqüência linear do documento. O sistema considera próximo nó desta versão o próximo nó da versão anterior exceto se esta contiver um button de visão caso em que o usuário deverá determinar o próximo nó manualmente. Caso chegue algum apontador para o nó que não seja parte da seqüência linear do documento (um button), o usuário será informado que a referência foi removida. Um nó removido pode voltar a existir apenas no caso de ser feita uma edição em uma versão anterior à sua remoção.

Ao abrir um documento, o sistema abre um menu com todas as versões existentes do documento e o usuário pode escolher qual versão (não necessariamente a mais atual) irá usar. No entanto, é bom ressaltar que qualquer edição em qualquer versão faz com que surja uma versão do nó com número igual à versão do documento mais um.

O sistema mantém uma *versão cronológica* dos documentos e nós. Cada alteração introduzida provoca o surgimento de uma versão mais atual. No exemplo da figura 8, a alteração do nó 7 implica na criação de sua terceira versão. No entanto, a versão do documento em pauta é

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

nó \ versão doc.	1	2	3	4	5	6	7	8	9	10
1	●				●			●		
2	●	●								
3	●	●	●		●			●		
4			●							
5	●			●		●	●			
6	●	●			●					
7	●			●						

Figura 8: Como as versões de Nós e Documentos são administradas.

a de número nove. Esta é que deve ser armazenada em cada nó para permitir que possamos ter acesso a versões anteriores do documento.

4.1.8 Visões

O conceito de *visão* de um documento é fundamental para tratar situações em que um mesmo documento pode ter mais de um uso, sendo que para cada uso apenas alguns trechos não são comuns.

Esta situação acontece, por exemplo, no caso de manuais de carros de uma determinada linha. Cada modelo tem algumas características únicas que o particularizam, mas a maior parte da documentação é comum a toda linha. No caso de um ambiente de desenvolvimento de software, a situação pode ocorrer quando se tem um software que irá rodar em diversos computadores. Nestes casos, seria ineficiente duplicar todas as informações já que os trechos particulares a cada *visão* são poucos. O conceito de visão está diretamente ligada aos *buttons* e *links* de visão.

Pode-se também criar um documento onde o leitor possa escolher o nível de detalhe de seu interesse do documento. Neste caso, determinados nós e *buttons* serão omitidos. No caso de uma empresa, pode-se criar um documento, onde deseja-se que informações técnicas não sejam vistas pelo pessoal de marketing e vice-versa.

Buttons de Visão

São buttons que, dependendo da escolha do usuário, levarão a uma visão diferente do documento. Poderão ser predefinidos pelo usuário no início da sessão de tal forma que a existência de mais de uma visão do documento se torne transparente. No caso do manual do carro, um button de visão selecionaria o modelo que deseja consultar. A partir desse momento, o hiperdocumento é visto como se tivesse sido escrito exclusivamente para este modelo.

Um nó pode conter no máximo um button de visão. Após a seleção de um button, o sistema abrirá a janela seguinte para a escolha da opção.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

4.1.9 Buttons de Decisão

Estes buttons são semelhantes aos buttons de visão, ou seja, são buttons onde o leitor deve fazer a escolha de uma alternativa. No entanto, não criam *visões* do documento. Portanto, buttons de decisão não podem ser pré-definidos.

Este tipo de button pode ser utilizado, por exemplo em uma aula de biologia, onde o leitor que está estudando o sistema respiratório pode escolher um mamífero, um réptil etc. Esta escolha fará com que ele tenha uma espécie de *visão local* do documento. Ao contrário de um button de visão, um nó pode conter diversos *buttons de decisão* .

4.1.10 HiperCorreio — Correio Eletrônico

O conceito de HiperCorreio é uma extensão do conceito de correio eletrônico normalmente existente em sistemas de computação. A diferença fundamental aqui é que nós, links e documentos podem ser enviados pelo Hiper Posto de Correio.

Uma carta eletrônica contém as seguintes informações:

```
< Id. remetente,
  Id. destinatario,
  data/hora que foi enviado,
  mensagem >
```

A mensagem é um documento e será retirada do sistema quando o destinatário acusar seu recebimento.

4.1.11 Button de Referência

Um *button de referência* é formado pelos seguintes dados:

```
< tipo (graphics/texto),
  conjunto de visoes,
  Id. no destino >
```

O conjunto de visões define para quais visões o *button* é ativo. Caso o button não seja ativo na visão, não será mostrado ao usuário. A definição de visões é feita na criação do documento.

Quando o button for intra documentos, todos os buttons do nó destino serão visíveis.

4.1.12 Palavras Chave

H permite a definição de palavras chave que podem ser localizadas de forma bem eficiente através de um comando ACHE. As keywords devem ser definidas por quem está escrevendo o hiperdocumento e servem tanto para localizar um texto como um *graphics* dentro do hiperdocumento. O sistema mantém uma tabela com as referências para maior eficiência.

4.1.13 Procure

H tem um comando PROCURE, válido apenas para textos. Com este comando é possível localizar *strings* dentro do hiperdocumento. Esta pesquisa é feita de modo seqüencial por todo hiperdocumento, não sendo portanto eficiente.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

4.1.14 Navegação

H vai dispor de um folheador gráfico (*browser gráfico*) de modo que o usuário tenha uma visão global do hiperdocumento e permita que se mova para qualquer nó sem passar pelos nós intermediários. O folheador gráfico manterá um mapa apenas da estrutura do documento formada pela seqüência linear e o button de decisão. Os buttons de referência não serão mostrados.

H armazena o caminho percorrido pelo usuário, que poderá ser *salvo*, caso a sessão seja interrompida e seja de seu interesse recordar o caminho por onde passou. O caminho também pode ser salvo para servir de roteiro para outros usuários. Este uso é particularmente útil no caso de aulas onde um professor pode armazenar um caminho mínimo que deseja que seus alunos percorram pelo hiperdocumento. Neste caso o sistema também deve permitir que o usuário faça seus próprios desvios e volte ao caminho original (armazenado pelo professor), no ponto que estava quando começou a navegar por conta própria, no momento que desejar.

5 Conclusão

Os sistemas de hipertextos trazem consigo a filosofia de armazenamento e recuperação de informações de objetos dispostos de forma não linear. Esta forma de armazenamento torna flexível e dinâmica a organização de documentos, onde as informações estão estruturadas de forma a se ter acesso "imediato" às mesmas através de links. O dinamismo com que a informação é tratada em sistemas de hipertexto, além da inter-relação e integração, cria uma espectativa de uso crescente destes sistemas.

A maior parte das bases de dados atuais se adequam ao modelo relacional e para se retirar informações destas bases o usuário deve ser bem treinado e ter uma boa idéida do que existe na base. Pela facilidade de uso que proporciona, os sistemas de hipertextos tem-se mostrado modelos mais adequados para usuários ocasionais pesquisar estas bases de dados.

Para o desenvolvimento de software, em particular, os sistemas de hipertexto são um importante recurso devido à integração não apenas de informação, como também de ferramentas utilizadas nas diversas fases do ciclo de vida do software. Além disso, o dinamismo com que a informação é tratada, através do particionamento e conexõesfacilita o "trabalho cooperativo", já que o desenvolvimento de software, em geral, é um somatório do esforço de vários integrantes de um projeto.

O desenvolvimento de software *fácil de usar* e *amigável* é uma tendência irreversível. E a hipermídia possui as características para melhor atingir estes objetivos. No futuro, as pessoas utilizando um sistema de hipermídia poderão entrar em bibliotecas, museus, bancos de dados, lojas etc. de dentro de suas próprias casas.

Uma outra verdade é que quanto mais simples de usar o software, mais difícil de ser desenvolvido. Mais uma vez hipermídia terá um papel fundamental. Por facilitar a documentação, controle de versão, controle do trabalho de grupos em um mesmo sistema, recuperação de informação etc., sistemas de hipermídia têm se mostrado ideais como ferramenta em ambientes de desenvolvimento de software.

O estudo de hipermeios ainda está no início no país. Não existe ainda nenhum produto no mercado. O sistema H é uma das primeiras tentativas de se fazer um protótipo de hipertexto no país e tem fornecido uma série de subsídios para a definição e desenvolvimento de um sistema mais completo e eficiente para um ambiente de desenvolvimento de software ora em estudos no DI-UFPE.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

Referências

[1] A. M. L. Vasconcelos, A. C. V. Melo e S. R. L. Meira: "Hex - Hipertexto como Suporte a Ambientes de Desenvolvimento de Software". Dep. de Informática, UFPE, Recife, PE, janeiro 1989.

[2] A. C. A. Sampaio: "Zc: Uma Notação para a Especificação de Sistemas Complexos". Dissertação de Mestrado, UFPE, novembro de 1988.

[3] S. R. L. Meira: "Ambiente para Desenvolvimento Rigoroso de Software". *Notes* **Nr. 1**, agosto 1988.

[4] P. K. Garg and W. Scacchi: "A Hypertext System to Manage Software Life Cycle Documents". *IEEE Software*, to be published, 1989.

[5] P. K. Garg and W. Scacchi: "On Designing Intelligent Software Hypertext Systems". Personal Communication.

[6] J. H. Walker: "Supporting Document Development with Concordia", *IEEE Computer*, Jan 1988, **pp 48-59.**

[7] G. Marchionini and B. Shneiderman: "Finding Facts vs. Browsing Knowledge in Hypertext Systems", *IEEE Computer*, Jan 1988, **pp 70-80.**

[8] N. M. Delisle and M. D. Schwartz: "Contexts -A Partitioning Concept for Hypertext", *ACM Transactions on Office Information Systems*, **Vol. 5, Nr. 2** , Apr 1987, **pp 168-186.**

[9] N. Delisle and M. Schwartz: "Neptune: A Hypertext System for CAD Applications",*ACM*, 1986, **pp 132-139.**

[10] R. H. Trigg and M. Weiser: "TEXTNET: A Network-Based Approach to text handling", *ACM Transactions on Office Information Systems*, **Vol. 4, Nr. 1**, Jan 1986.

[11] F. M. Akscyn: "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems". *Communications of the ACM*, **Vol. 31 Nr. 7**, July 1988.

[12] N. Meyrowitz: "Intermedia: The Architecture and Construction of on Object-Oriented Hypermedia System and Aplications Framework", *ACM*, Sep 1986.

[13] M. L. Begeman and J. Conklin: "The Right Tool for the Job" *BYTE*, Oct 1988.

[14] A. Newell, D. L. McCracken, G. Robertson and R. M. Akscyn: "ZOG and the USS Carl Vinson", Carnegie-Mellon University, 1982.

[15] P. J. Brown: "Turning ideas into products: the Guide system". Personal Communication.

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

[17] J. Kelner, A. L. C. Cavalcante e A. Pardo: "LindA: Uma linguagem de Autoria Automátfca para Hipertexto". III Simpósio Brasileiro de Engenharia de Softwate 1989.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor