Id: Um Sistema de Hipertexto Configurável e Orientado a Objetos para Integrar os Documentos do Software

Renato Silva Cabral* Silvio Lemos Meira

Departamento de Informática
Universidade Federal de Pernambuco
Caixa Postal 7851, 50732-970, Recife, PE, Brasil
e-mail:{rsc,srlm}@di.ufpe.br

Sumário

A falta de integração dos documentos produzidos durante o Ciclo de Vida do Software (CiViS) é um dos problemas enfrentados para fazer a manutenção do mesmo. A tecnologia de Hipertexto tem sido muito explorada na construção de ferramentas CASE para a integração de documentos. Isto acontece pela facilidade de relacionar documentos através de referências (*links*).

Este trabalho propõe uma arquitetura orientada a objeto de hipertexto para integração de documentos do CiViS. Esta arquitetura é configurável no sentido de estruturar os documentos independentemente da metodologia usada no desenvolvimento e por permitir que os conteúdos dos nós sejam manipulados por ferramentas externas. Para isso propõe-se um protocolo de comunicação com estas ferramentas para que se possa definir âncoras nos conteúdos manipulados por elas.

Utilizando esta proposta, um sistema de hipertexto foi especificado formalmente utilizando a linguagem MooZ (Modular Object Oriented Z). Então, um protótipo deste hipertexto, denominado Id, foi construído e os resultados dos primeiros testes mostram que os objetivos da arquitetura estão sendo alcançados.

Abstract

One of the hardest problems of mainthinking software is the lack of integration of the documentation produced throghout the rest of the software life cycle. Hypertext technology has been widely explored in CASE tools to dea with some of the problems of integration, due to the capabilities associated to linking paradigm.

This work proposes a configurable, methodology independent, architecture for document integration, allowing for external tools to handle node contents and anchor definition through a communication protocol.

MooZ was used to produce a formal specification of a hypertext system based on the architecture, a prototype of which was built. Testes show that the basic goals of the project have been reached.

*Atualmente Analista de Sistemas da Fóton Informática e Serviços Ltda.- Brasília/DF.

1 Introdução

Um dos prismas através do qual a crise de software pode ser vista diz respeito à documentação do software. Independente do modelo de desenvolvimento seguido por quem constrói um software, é extremamente necessário que se registre o máximo de informação sobre este desenvolvimento. Estas informações são tão ou até mais importantes que o código final implementado. As leis de Lehman [19] ressaltam como característica inerente ao software a evolução. Estas leis garantem que um software em uso num ambiente real está em mudança contínua, acompanhado de um constante aumento de complexidade. A documentação vai permitir que estas mudanças sejam feitas para corrigir, estender, refinar ou fazer qualquer outro tipo de manutenção. Através da documentação bem feita é possível que o software seja analisado nos diversos níveis de abstração (por exemplo, especificação, projeto, implementação), ou por pessoas que tenham diferentes visões, como o projetista, o gerente, o usuário, o programador, etc.

Outro aspecto importante é o relacionamento entre os diversos tipos e níveis da documentação, que normalmente não é registrado. O que geralmente acontece é o registro independente de cada fase do desenvolvimento. Este isolamento provoca muitas vezes a inconsistência da documentação de uma fase com a de outra, ao final do desenvolvimento. Às vezes, mudanças são feitas e as revisões necessárias não são realizadas nos documentos relacionados, por falta de registro destas relações. O controle da consistência, associado às versões isoladas dos documentos, é portanto dificultado.

Hipertexto e Integração de Documentos

Hipertexto é uma maneira de estruturar documentos de forma não linear. Os documentos são divididos em nós que podem estar ligados uns com os outros através de referências (*links*). As referências podem ligar pontos específicos dentro do conteúdo do nó que são chamados de âncoras. O conteúdo de um nó pode ser de vários tipos, tais como texto, gráfico, imagem, som, código executável, etc. Esta forma de estruturação dá uma grande flexibilidade para que se faça o relacionamento entre documentos que não possuem uma estrutura de relacionamento bem definida.

O relacionamento entre os documentos do CiViS pode se beneficiar da tecnologia de hipertexto justamente porque este relacionamento não tem uma forma muito bem definida, além dos documentos poderem ser de vários tipos. A ligação entre os documentos através de referências pode ser explorada não somente como um mecanismo de navegação, mas também como uma forma de associar mais informação a estes relacionamentos. Com isso é possível registrar de forma mais rica o significado do relacionamento existente e, a partir daí, examinar com maior precisão a influência de mudanças de um documento sobre outro.

Neste trabalho é descrito a experiência do desenvolvimento de um sistema de hipertexto para integrar os documentos do software cuja a principal característica abordada é a de ser configurável. A arquitetura de sistema foi modelada utilizando as tecnologias de orientação a objetos (OO) e de linguagens de especificação formal, no caso MooZ. Um protótipo foi então produzido e testado.

A próxima seção mostra como hipertexto tem sido usado na automação do CiViS, o que fornece base para definição dos requisitos de Id. A seção seguinte descreve a especificação de Id, mostrando em mais detalhe o modelo OO. Então são descritos alguns aspectos relativos à implementação do protótipo produzido e testes. Na conclusão, destaca-se as principais contribuições do trabalho e as perspectivas de extensões.

2 Hipertexto e Automação do Ciclo de Vida do Software

A utilização da tecnologia de hipertexto na automação do CiViS tem sido muito grande, como descrito em [6, 9, 7, 16, 1]. Dentre as motivações para o uso de hipertexto citadas nestes trabalhos, pode-se destacar as seguintes:

- Utilização da flexibilidade de ligação dos documentos, realizada através das referências, para fazer um melhor controle sobre documentação de todo o CiViS buscando maior consistência, completitude, capacidade de traçabilidade e eficiência na recuperação de informações.
- A possibilidade de utilizar diferentes tipos de documentos, que é uma característica dos documentos manipulados no CiViS.

A grande parte destes trabalhos utiliza hipertexto para relacionar documentos envolvidos durante todo o CiViS. Dentre esses, um dos que mais se destacam é DIF (*Documents Integration Facility*) [9], que foi uma das principais motivações e inspirações do presente trabalho. DIF foi produzido com os seguintes objetivos:

- Prover uma interface com as ferramentas que manipulam os documentos utilizados no CiViS.
- Prover mecanismos para controlar a consistência e completitude dos documentos, assim como a capacidade de traçabilidade entre eles.
- Incentivar o reuso de documentos já produzidos em outros projetos.
- Capacitar a documentação de software de grande porte.
- Dar suporte à construção de software composto por múltiplos projetos.

Os resultados apresentados mostram que a utilização da tecnologia de hipertexto na implementação de DIF atenderam os objetivos citados.

Há, entretanto, outros trabalhos que utilizam hipertexto para o relacionamento de documentos específicos de uma fase do CiViS, como por exemplo ForMooZ [16]. Esta é uma ferramenta, produzida como trabalho de mestrado no Departamento de Informática da UFPE, que utiliza hipertexto como forma de relacionar as informações dentro de uma especificação formal na linguagem MooZ [13]. Por explorar um tipo específico de documento, a ferramenta domina a estrutura sintática e semântica dos mesmos. Com isso, ForMooZ é capaz de implementar uma geração automática de referências a partir da compilação das especificações. Deste modo, consegue-se manter as referências atualizadas depois de alterações nas especificações.

O estudo destes trabalhos deram base para definir os requisitos de um sistema de hipertexto com o objetivo de integrar os documentos produzidos no CiViS, descrito na próxima seção.

2.1 Requisitos de Id

Baseado no estudo das características básicas de hipertexto e sua aplicação para construção de ferramentas CASE, vários requisitos foram identificados para integrar os documentos do CiViS através de um sistema de hipertexto. Dentre eles, os considerados mais importantes foram:

- Suporte a Ambiente Distribuído: o desenvolvimento de software de grande porte é feito por grandes equipes. Estações de trabalho em rede local é a tendência dos ambientes de desenvolvimento.

- Suporte a Trabalho Cooperativo: uma equipe de desenvolvimento deve trabalhar de forma que cada componente possa cooperar com o outro na produção dos documentos do software.

- Configurável: As metodologias usadas no desenvolvimento de um software, tanto a nível de processo quanto de ferramentas utilizadas, estão em constante adaptação. Além disso, não há uma metodologia padrão que seja amplamente usada. Estas variações têm reflexos na estruturação e nos tipos de documentos produzidos. Uma ferramenta de integração de documentos deve ser capaz de se adaptar junto com a metodologia e demais ferramentas utilizadas.

- Suporte a Ambiente Multi-Projeto: É muito comum que o desenvolvimento de um software esteja relacionado com alguns outros projeto. Deste modo, pode-se relacionar os documentos de software/projetos diferentes.

- Relacionamento dos Documentos: A documentação de um software deve ser feita de modo que se registre o relacionamento dos vários documentos que a compõem. Este registro dá a possibilidade de alcançar a característica de traçabilidade, onde pode-se saber, por exemplo, qual o programa que implementa determinada funcionalidade da especificação.

- Controle de Versões: o desenvolvimento de software é um processo dinâmico onde os documentos sofrem muitas mudanças durante todo o ciclo. O controle de versões dos documentos é importante, assim como o registro da causa da mudança de versão, para que estes documentos realmente auxiliem na sua própria manutenção.

- Geração de Documentação Impressa: a tecnologia atual ainda não conseguiu substituir o papel como o principal meio de manipulação de documentos. Portanto é necessário que se permita a geração automatizada de documentação para ser impressa, a partir do formato eletrônico.

- Visões: a documentação de um software é manipulada por diferentes pessoas que possuem necessidades distintas, de acordo com a função que desempenham no ambiente de desenvolvimento.

- Consistência: uma situação muito comum no CiViS é ter programa(s) executável(is) com uma documentação desatualizada e inconsistente. Uma ferramenta de integração de documentos deve prover mecanismos para identificar tais inconsistências.

- Informações Complementares: no processo de desenvolvimento de software é muito comum se fazer anotações, associadas aos documentos, para explicar ou esclarecer algo sobre o mesmo.

Acredita-se que para construir um sistema de hipertexto que atinja o objetivo mais amplo de integrar a documentação do CiViS, todos os requisitos apresentados devem ser atendidos. Neste trabalho escolheu-se o requisito de ser configurável para ser atendido na primeira versão de Id, por ser este um dos mais básicos sobre o qual os outros podem ser adicionados, desde que se projete uma arquitetura fácil de ser estendida.

3 Especificação de Id

3.1 Bases da Especificação

A especificação de **Id** tem como objetivo principal atingir a característica de ser um hipertexto configurável. Para o aspecto de organização dos documentos, utilizou-se a forma hierárquica, baseada nas arquiteturas dos sistemas de hipertexto Neptune ([4]). Neste mecanismo tem-se o conceito de nó de contexto que funciona como um agrupador de outros nós e não possui documentos associados a ele. Este é um dos mecanismos de abstração de hipertexto, chamado agregação (definido em [10]) e com o qual costuma-se lidar com documentos impressos em papel. É o chamado formato de livro (divide documentos em capítulos, seções, páginas, parágrafos, etc.) que segundo [14] é uma forma de documentar programas que tem aumentado a rapidez no entendimento dos mesmos para fazer manutenções.

O segundo aspecto a ser tratado é permitir a utilização de documentos manipulados por diferentes ferramentas. A documentação de um software é composta por vários tipos de documentos, cujos conteúdos são estruturados, apresentados e editados de várias formas. Por exemplo, existem documentos do tipo texto, gráfico, código executável, código de processador de texto, dados específicos de ferramentas, código de programa, etc. Cada um destes tipos de documentos é manipulado por ferramenta específica e que não "sabe" da existência de um sistema de hipertexto. O objetivo da arquitetura proposta para **Id** é justamente permitir que todos estes tipos de documentos possam ser ligados através de referências entre nós.

Deste modo, tornou-se necessário ter uma máquina de hipertexto independente das estruturas dos conteúdos dos documentos. Os modelos abstratos Dexter ([11]) e HAM([4]) propõem arquiteturas que vêm de encontro a esta necessidade, onde a máquina de hipertexto é uma camada separada do armazenamento e da apresentação dos nós. Inspirado nestas idéias, propõem-se para **Id** uma máquina de hipertexto onde os conteúdos dos nós são atributos dos mesmos, mas cuja as estruturas são manipuladas somente pelas camadas de armazenamento e apresentação das mesmas. A diferença da proposta de **Id** para estas outras arquiteturas está justamente no desconhecimento da estrutura de cada documento. Com isso, as referências foram definidas entre nós e não entre regiões específicas do conteúdo do documento (âncoras).

Com esta proposta é possível que se tenha referências entre nós cujos conteúdos podem ser manipulados, por exemplo, pelo editor de texto "emacs" ou pelo editor gráfico "idraw". Entretanto com este tipo de arquitetura, a ligação de documentos utilizando âncora tornou-se um problema a ser resolvido. O modelo de hipertexto PROXHY ([12]) propõe justamente um protocolo de comunicação entre as ferramentas externas e a máquina de hipertexto para manipulação de âncoras. Incorporando esta característica a arquitetura de **Id** é possível que se tenha ferramentas que possam ser adaptadas para trocar mensagens com o sistema de hipertexto para manipulação de âncoras. A vantagem no caso do protocolo proposto para **Id** (definido na seção 3.2) em relação a PROXHY é que neste último as referências só podem ser definidas para ligar nós cujos os conteúdos são manipulados por ferramentas que conhecem o protocolo. No caso de **Id**, caso a ferramenta conheça o protocolo ela o utiliza para manipular âncoras, caso contrário as referências são criadas somente entre nós. Isso permite que o atual ambiente de desenvolvimento seja incrementado com um sistema de integração sem que seja necessário abandonar as ferramentas utilizadas.

A próxima seção descreve as principais características do modelo orientado a objeto proposto para **Id**.

3.2 Modelo Orientado a Objeto

Baseado nos aspectos discutidos na seção anterior e no estudo das características principais dos sistemas de hipertexto, foi feita uma especificação informal das características de Id (descritas em detalhe em [2]).

Com o objetivo de produzir um projeto com facilidade de manutenção e extensão, além do aumento da reusabilidade dos componentes definidos, utilizou-se o paradigma de orientação a objeto (OO). Este paradigma possui conceitos que têm possibilitado alcançar tais características, conforme mostra [17]. A metodologia utilizada na modelagem foi OMT [15] de Rumbaugh et al., influenciado pela experiência adquirida e descrita em [3].

O resultado da modelagem OO foi a definição das classes e seus interrelacionamentos. As classes foram agrupadas em subsistemas de acordo com sua funcionalidade, definindo a arquitetura de Id. A figura 1 mostra o relacionamento cliente-servidor dos subsistemas, de acordo com a notação proposta pela metodologia.

Figura 1: Arquitetura de Id - Divisão em Subsistemas

As características principais de cada subsistema são:

- Interface: é responsável pela apresentação das janelas que mostrarão os dados do hipertexto.
- Comunicação Externa: coopera com o de interface e é responsável pela visualização e edição de conteúdo através de ferramentas externas e pela comunicação com essas ferramentas.
- Máquina de Hipertexto: como o nome sugere, gerencia os dados dos nós, referências e âncoras que compõem o hipertexto.
- Gerência de Tipo: dá suporte à máquina de hipertexto, controlando os tipos de documentos cadastrados por Id. Os tipos de documentos definem a forma de editar, visualizar e armazenar os conteúdos dos nós de documento do hipertexto.
- Conteúdo de Nó: é servidor do subsistema máquina de hipertexto, com a responsabilidade de registrar os dados sobre os conteúdos dos nós de acordo com o tipo.

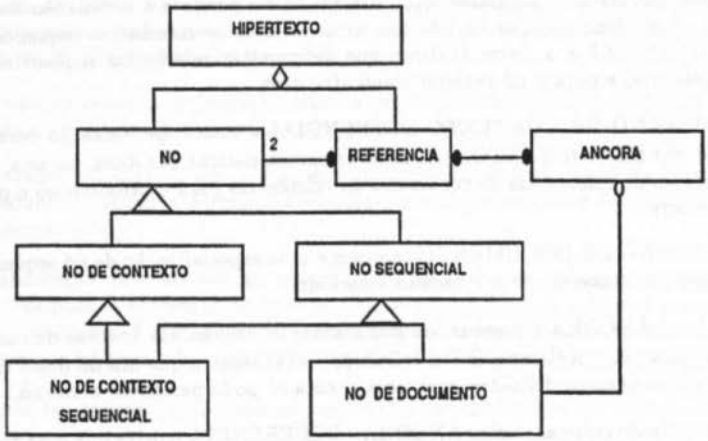

Figura 2: Diagrama de Classes do Subsistema de Máquina de Hipertexto

A figura 2 mostra o diagrama de classes do subsistema de máquina de hipertexto. Sobre este diagrama destaca-se as seguintes observações:

- A classe HIPERTEXTO é responsável pela gerência dos objetos das classes que a compõem.
- Conforme mostra a relação todo-parte da figura, as partes do hipertexto são nós e referências.
- A classe NÓ define as características básicas de um nó, que são o nome, o conjunto de referências para outros nós (chamadas de referências gerais) e uma anotação.
- A classe REFERÊNCIA define como atributo a identificação dos dois nós que liga, que no caso de **Id** são sempre bidirecionais. Entretanto, para domínios específicos, um atributo para indicar o sentido poderia ser adicionado. Isso foi definido para **Id** porque estas referências são usadas para mostrar que um documento influencia no outro e que neste caso esta influência é inerentemente nos dois sentidos.
- A associação entre as classes NÓ e REFERÊNCIA mostra que uma referência liga dois nós, mas um nó pode possuir muitas referências.
- A relação de herança da figura mostra a especialização da classe NÓ para as classes NÓ DE CONTEXTO e NÓ SEQUENCIAL.
- A classe NÓ DE CONTEXTO define referências para os nós que estão dentro do seu contexto (chamadas de referências de contexto). As instâncias desta classe são nós que representam a raiz da hierarquia de nós.
- A organização dos documentos no formato livro é feita pelas estruturas hierárquica e seqüêncial. A hierarquia possibilita dividir um livro em capítulos, capítulos em

seções, seções em parágrafos, etc., e a seqüência permite a ordenação destas partes. A seqüencialização de nós são feitas através de referências seqüenciais. NÓ SEQUENCIAL é a classe abstrata que define estas referências seqüenciais para o próximo nó e para o nó anterior como atributos.

- A classe NÓ DE CONTEXTO SEQUENCIAL é uma especialização dessa última e de NÓ DE CONTEXTO, herdando as características das duas, ou seja, possui o conjunto de referências de contexto e as referências para o próximo nó e para o nó anterior.
- A classe NÓ DE DOCUMENTO também é uma especialização de nó seqüencial que possui um conteúdo de documento associado.
- A classe ÂNCORA é responsável por manter os valores das âncoras de cada nó. A associação com a classe NÓ DE DOCUMENTO mostra que um nó deste tipo pode ter várias âncoras definidas, mas uma âncora só pode pertencer a um nó.
- A associação entre as classes ÂNCORA e REFERÊNCIA mostra que uma âncora pode estar ligada a várias referências (associadas ao mesmo nó da âncora) e vice-versa. A forma de implementar esta relação deve ser ressaltada porque uma referência não tem conhecimento das âncoras relacionadas com ela, mas as âncoras, sim. Isto acontece para que mudanças no conteúdo de um documento que afetem a definição de âncoras relacionadas não façam com que a ligação entre dois nós torne-se inválida.

Figura 3: Diagrama de Classes do Subsistema de Gerência de Tipo

Associado a um nó de documento existe um tipo de documento que define a forma de armazenamento, visualização e edição do seu conteúdo. A figura 3 mostra o relacionamento das classes do subsistema de gerência de tipos, onde pode-se destacar o seguinte:

- A classe TIPO DE DOCUMENTO é composta (relação todo-parte da figura) pelas classes TIPO DE CONTEÚDO, TIPO DE VISUALIZAÇÃO e TIPO DE EDIÇÃO.

- A classe TIPO DE CONTEÚDO define como o conteúdo do nó de documento é armazenado. Nesta primeira versão só considerou-se o armazenamento através de um ou mais arquivos no sistema operacional Unix.

- O tipo de visualização e edição definem quais as ferramentas que farão tais operações para o conteúdo do documento relacionados com o nó.

- A classe GERENTE DE TIPOS é composta (relação todo-parte da figura) por um conjunto de tipos e é responsável pela inclusão, alteração e remoção de tipos ao sistema.

O subsistema de comunicação externa, responsável pela ativação e desativação de ferramentas para a apresentação de conteúdo dos documentos, utilizará o protocolo de comunicação com as ferramentas de visualização conforme descrito abaixo.

No sentido Id para a ferramenta são quatro as mensagens definidas pelo protocolo:

- "mostrar âncora", cujo parâmetro é o valor de uma âncora recebido de Id.

- "esconder âncora", cujo o parâmetro é o valor de uma âncora.

- "esconder todas as âncoras", sem parâmetro.

- "terminar processo", informando para a ferramenta terminar o processo.

No sentido ferramenta para Id as mensagens são as seguintes:

- " âncora selecionada", cujo parâmetro é o valor da âncora selecionada pelo usuário.

- "seguir âncora", também com o valor da âncora como parâmetro.

- "processo terminado", informando que a ferramenta terminou a execução.

Através deste protocolo é possível associar ou dessassociar uma âncora a uma ou mais referências. Desta maneira, uma âncora pode ativar indiretamente (através da referência) um ou mais nós para serem abertos através da mensagem "seguir âncora".

3.3 Especificação Formal

As classes definidas pela arquitetura de Id foram refinadas e tiveram suas características (atributos e operações) especificadas formalmente utilizando a linguagem de especificação formal MooZ [13]. MooZ (Modular Object Oriented Z) é uma extensão da linguagem Z para incluir os conceitos de modularização através de classes, herança e polimorfismo usados por OO. A especificação dos subsistemas de máquina de hipertexto, comunicação externa, gerência de tipo e conteúdo de nó é apresentada em detalhes em [2].

Baseado neste modelo, foi construído um protótipo de Id. A próxima seção descreve alguns aspectos relevantes da implementação e teste deste protótipo.

4 Protótipo de Id

4.1 Implementação

A plataforma de desenvolvimento do protótipo foi estações de trabalho SUN com sistema operacional Unix da SUN. A linguagem de programação OO escolhida para implementar as classes definidas foi Sather que possui as características fundamentais deste paradigma. Além disso, Sather busca o aumento da reusabilidade, através de classes parametrizadas, e portabilidade, através da geração de código na linguagem C.

A interface foi implementada na linguagem C utilizando a ferramenta *Open Windows Developer's Guide* que gera programas para o sistema de interface *Open Windows*, ambos disponíveis para as estações SUN.

O armazenamento dos objetos das classes definidas foi feito através do GOD (gerenciador de objetos distribuídos), implementado como trabalho de mestrado na UFPE [7]. GOD dá persistência a objetos da linguagem Sather utilizando a filosofia cliente-servidor.

Estas escolhas para implementação foram feitas porque atendiam aos requisitos de Id, além de terem sido influenciadas pelas experiências bem sucedidas nas implementações das ferramentas ForMooZ [16] e TOMRULES [18], que utilizaram as mesmas ferramentas e plataforma.

A implementação desta primeira versão de Id foi muito beneficiada pela utilização da linguagem de especificação formal MooZ, que permitiu que se tivesse uma definição bem precisa e estável do que seria codificado. Isto foi refletido no tempo relativamente curto de codificação de todo o protótipo produzido, que foi, aproximadamente, seis semanas. O código final definiu aproximadamente cinqüenta classes em Sather (oito mil linhas de código) e reusou outras cinqüenta classes da biblioteca de Sather. A definição da interface produziu vinte mil linhas de código C cuja grande parte foi gerada automaticamente pela ferramenta *Guide*.

Por fim, foi produzido um protótipo estável, conforme mostraram os testes comentados na próxima seção.

4.2 Testes

O primeiro projeto cujos os documentos foram incluídos no sistema de hipertexto de Id foi o da própria documentação de Id.

Os documentos produzidos durante o desenvolvimento usaram várias ferramentas, como o editor de texto "emacs", o editor gráfico de arquivos *post script* "idraw", o visualizador de arquivos *post script* "ghostview", além da ferramenta de definição de interface "guide". Todas essas ferramentas foram usadas para definição de tipos de documentos, já que todas possuíam os requisitos de:

- Armazenar os conteúdos dos documentos em arquivos Unix.
- Poderem ser chamadas passando como parâmetro o nome destes arquivos para serem manipulados pelas mesmas.

Feito o cadastramento dos tipos, foram definidos os nós de contexto, contexto seqüencial e de documento que compunham a documentação já feita para Id. Na figura 4 é mostrado a interface de um nó de documento cujo conteúdo é visualizado através da ferramenta "ghostview". O tipo de documento deste nó foi definido como `Post script`. Este tipo é especificado através da janela de definição de tipos de documento, mostrada na figura 5.

Para testar o uso do protocolo de âncoras, utilizou-se a ferramenta FEGER, que foi desenvolvida como trabalho de mestrado na UFPE [8]. Esta é uma ferramenta gráfica

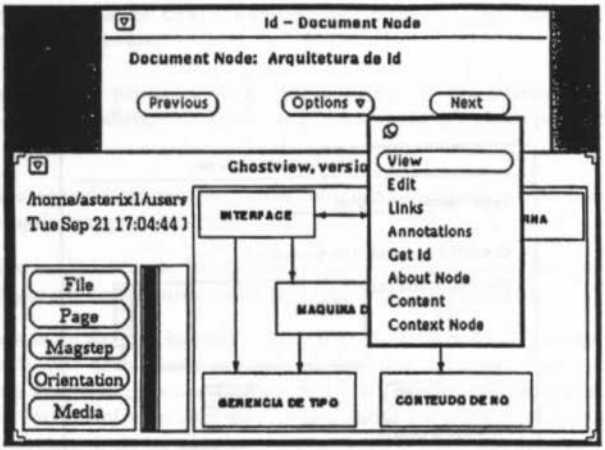

Figura 4: **Janela de Nó de Documento** - *Mostra as opções de operações disponíveis para um nó de documento que neste caso tem seu conteúdo o desenho da arquitetura de* **Id**.

para edição de diagramas entidade-relacionamento estendidos e tradução destes diagramas em modelos específicos de sistemas de bancos de dados. A disponibilidade dos fontes de FEGER possibilitou a realização das alterações necessárias para implementação das rotinas de definição de âncoras e do protocolo de comunicação de **Id**.

Os testes mostraram que a definição de nós, referências e âncoras em **Id** atende às características essenciais de um sistema de hipertexto. Quanto a característica de ser configurável, foi possível constatar através dos testes que a organização dos nós foi feita conforme a metodologia utilizada e que as ferramentas utilizadas durante o desenvolvimento também poderam ser acopladas ao hipertexto. Além disso, o protocolo de âncoras permitiu que uma ferramenta fosse acrescentada e fizesse a manipulação de âncoras conforme o especificado.

5 Conclusão

As contribuições que podem ser destacadas com a realização deste trabalho são as seguintes:

- Definição da arquitetura de um hipertexto configurável para integrar documentos do CiViS, no sentido de permitir a estruturação de documentos independentemente da metodologia e a integração das ferramentas utilizadas na manipulação destes documentos.
- Especificação formal orientada a objeto da arquitetura proposta na linguagem MoZ.
- Implementação e testes de um protótipo com as características especificadas.

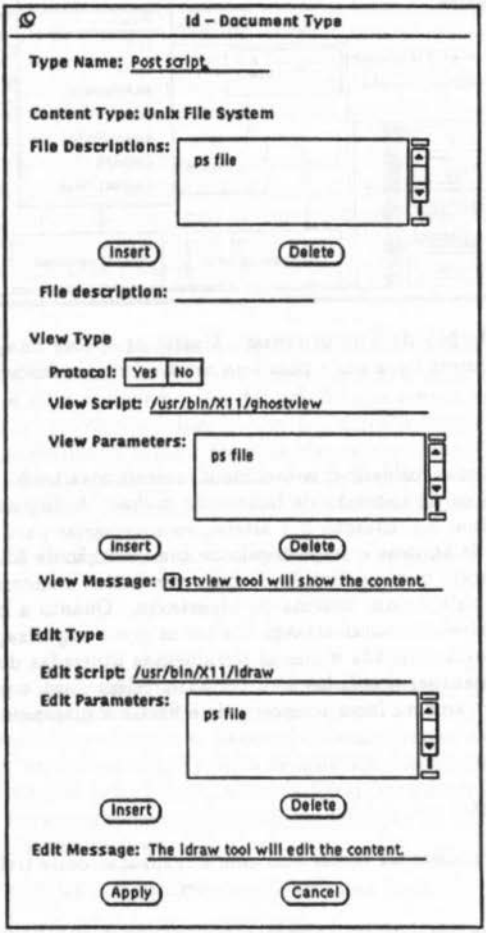

Figura 5: **Janela de Definição de Tipos** - *Define os dados do tipo de documento, especificando os tipos de conteúdo, visualização e edição.*

- Experiência na utilização de métodos formais em conjunto com o paradigma de orientação a objeto no CiViS, que tem sido tema de pesquisas recentes na área de engenharia de software.

Com estes trabalho procurou-se montar a base para um sistema de hipertexto que pode evoluir em várias direções. Dentre as propostas de extensões para **Id**, pode-se citar as seguintes:

- Implementação dos demais requisitos definidos para que **Id** possa atingir plenamente o objetivo de integrar documento do CiViS.
- Extensões multimídia de **Id** através do desenvolvimento de ferramentas que manipulem tipos de documentos como som, gráficos, imagens, etc.
- Mecanismos de navegação mais sofisticados devem ser providos para facilitar o percorrimento dos nós. Neste sentido, no trabalho de mestrado descrito em [5], foi feita uma extensão para arquitetura de **Id** utilizando uma teoria de modelos cognitivos que incrementou as informações associadas aos nós e referências, possibilitando novas alternativas de navegação.

Com o presente trabalho espera-se estar contribuindo para as pesquisas nas área de hipertexto e de integração de documentos do CiViS. Os resultados do protótipo produzido mostraram que as idéias exploradas pela arquitetura de **Id** são factíveis. Além disso, a proposta de uma arquitetura configurável orientada a objeto tem facilitado as extensões feitas para **Id**.

6 Agradecimentos

Esse trabalho foi desenvolvido com o apoio financeiro do CNPq e da FACEPE.
Agradecimentos especiais a Márcia Cardador pela revisão e impressão deste artigo.

Referências

[1] C.G. Alves and M.R.S. Borges. HyFor - Uma Ferramenta de Apoio a Manutenção de Software Científico. In *VII Simpósio Brasileiro de Engenharia de Software*, October 1993.

[2] Renato Silva Cabral. **Id**: Um Sistema de Hipertexto Configurável Orientado a Objeto para Integrar Documentos do Ciclo de Vida do Software. Master's thesis, Depto. de Informática - Universidade Federal de Pernambuco, March 1994.

[3] R.S. Cabral, S.A. Jansen, P.P. Carreiro, and J.B. Castro. Utilização da Metodologia OMT na Construção de Ferramentas CASE. In *VII Simpósio Brasileiro de Engenharia de Software*, October 1993.

[4] B. Campbell and J.M. Goodman. HAM: A General Purpose Hypertext Abstract Machine. *Communications of ACM*, 31(7), July 1988.

[5] Márcia Mendonça Cardador. Modelos de Categorização Aplicados a Sistemas de Hipertexto: Uma Perspectiva Cognitiva. Master's thesis, Depto. de Informática - Universidade Federal de Pernambuco, July 1994.

[6] J.L. Cybulski and K. Reed. A Hypertext Based Software Engineering Environment. *IEEE Software*, March 1992.

[7] J.C. Ferrans, D.W. Hurst, M.A. Sennett, B.M. Connot, W. Ji, P. Kajka, and W. Ouyang. A HyperWeb: A Framework for Hypermedia-Based Environments. In *ACM SigSoft'92: Fifth Symposium on Software Development Environment (SDE5)*, 1992.

[8] Marum S. Filho. Uma ferramenta gráfica de apoio ao projeto de banco de dados. Master's thesis, Departamento de Informática - UFPE, 1994. (Em Preparação).

[9] Pankaj K. Garg and Walt Scacchi. A Hypertext System to Manage Software Life-Cycle Documents. *IEEE Software*, pages 90–98, May 1990.

[10] P.K. Garg. Abstraction Mechanisms in Hypertext. *Communications of ACM*, 31(7), July 1988.

[11] F. Halasz and M. Schwartz. The Dexter Hypertext Reference Model. In J. Molina, D. Benigni, and J. Baronas, editors, *The Hypertext Standartization Workshop*, January 1990.

[12] C.J. Kacmar and J.J. Leggett. PROXHY: A Process-Oriented Extensible Hypertext Architecture. *ACM Transactions on Information Systems*, 9(4):339–419, October 1991.

[13] S.R.L. Meira and A.L.C. Cavalcanti. The mooz specification language - version 0.4. Technical Report ES/1.92, ProTeM-CC-NE, January 1992.

[14] P.W. Omam and C.R. Cook. Typographic Style is More than Cosmetic. *Communications of the ACM*, 33(5):506–520, May 1990.

[15] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design*. Prentice-Hall International Editions, 1991.

[16] C.S. Santos. Formooz: Um ambiente multi-usuário baseado em hipertexto de suporte à construção de especificações formais orientadas a objeto. Master's thesis, Departamento de Informática - UFPE, July 1992.

[17] J. Scholtz, S. Chidamber, R. Glass, A. Goerner, M. Rossan an M. Stark, and I. Vessey. Object-Oriented Programming: The Promise and the Reality. *J. Systems Software*, 1993.

[18] André F.C. Silva. Tomrules - um monitor de eventos, regras e gatilhos em um ambiente orientado a objeto. Master's thesis, Departamento de Informática - UFPB, March 1993.

[19] Ian Sommerville. *Software Engineering*. Addison-Wesley, third edition, 1989.