HyFor - Uma ferramenta de apoio a manutenção de software científico

Carla Gama Alves
Programa de Engenharia de Sistemas/COPPE

e

Marcos Roberto S. Borges
Núcleo de Computação Eletrônica
e
Departamento de Computação
Universidade Federal do Rio de Janeiro

NCE/UFRJ
Caixa Postal 2324
20001-970 Rio de Janeiro
e-mail: mborges@nce.ufrj.br

RESUMO

Há uma grande quantidade de software científico desenvolvida por engenheiros e técnicos não especializados em computação sendo utilizado em empresas, especialmente de engenharia. A grande maioria desses softwares foi desenvolvida sem nenhuma técnica de engenharia de software e possui o mínimo de documentação, provocando um alto custo de manutenção. Um ambiente voltado especificamente para uma maior produtividade no trabalho de manutenção desse tipo de software é descrito nesse artigo. O paradigma de hipertexto e a associação de relatórios de avaliação de qualidade são as principais características desse ambiente.

ABSTRACT

There is a great amount of scientific software developed by engineers and non-specialized technicians being used on companies, especially engineering companies. The great majority of this software was built without applying any software engineering techniques and has a minimum documentation, causing an increasing cost of maintenance. A special environment designed specially for achieving higher productivity on the maintenance of this type of software is described in this article. The hypertext paradigm and the association of quality evaluation reports are the major aspects of this environment.

I - INTRODUÇÃO

Ambientes de Desenvolvimento de Software Científico são notadamente mais resistentes à implementação de técnicas de engenharia de software do que outros ambientes tradicionais de sistemas de informação. Uma das causas está no fato de que os desenvolvedores e implementadores do software científico são principalmente os próprios engenheiros, que estão distantes das técnicas mais atuais da engenharia de software.

Outra causa, não menos importante, é que o ferramental existente não é adequado à solução dos problemas encontrados nesse tipo de ambiente. Pouca atenção é dada na engenharia de software à solução desses problemas específicos do ambiente científico, concentrando-se a maior parte do estudo em problemas encontrados em ambientes tradicionais.

Por outro lado, a quantidade de software científico em utilização e o conseqüente tempo gasto na sua manutenção faz crer que a crise de software nesse ambiente seja tão ou mais aguda que nos ambientes de sistemas de informação tradicionais. Urge, portanto, desenvolver soluções apropriadas voltadas para a especificidade do problema, levando-se em consideração, principalmente, as diferenças de cultura e formação dos profissionais envolvidos.

Sabe-se que grande parte do software científico foi desenvolvido sem qualquer metodologia ou técnica de construção de sistemas. Na maior parte dos casos os programas, geralmente em FORTRAN, servem ao mesmo tempo de especificação, implementação e documentação do sistema. Em muitos casos um programa é o próprio sistema.

Na engenharia de software tradicional grande esforço tem sido feito no sentido de desenvolver técnicas, modelos e ferramentas de suporte ao desenvolvimento de sistemas. Importantes propostas estão voltadas para implementação de métodos a partir da especificação do problema, refletindo, a partir daí, nas demais fases do processo de desenvolvimento.

Acreditamos que no caso do software científico, especialmente, este enfoque tem pouca chance de sucesso, seja em função da formação das pessoas, eminentemente prática, seja pela falta completa de cultura nesse sentido. Assim, defendemos que a inserção de métodos nesse ambiente se dê da forma "bottom-up", ou seja, a partir de técnicas de programação evoluindo até a especificação.

Este artigo descreve um ambiente de apoio à última etapa do processo de desenvolvimento, ou seja, a manutenção de programas. Acreditamos que com esse enfoque iremos gradativamente convencer o especialista da necessidade de maior rigor na construção do software, atingindo, eventualmente, a etapa inicial de especificação.

Utilizamos técnicas de hipertextos, consideradas de fácil assimilação para usuários não especializados, para organizar as diversas informações do programa, tais como o próprio código fonte, comentários e versões. Como benefício adicional e como forma de convencer o usuário a utilizar o ambiente, incluímos facilidades para avaliação de qualidade e de complexidade do programa. Tanto a geração do hiperdocumento, como a regeneração do código fonte a partir deste, são realizadas de forma automática, introduzindo o mínimo de "overhead" no trabalho do especialista.

II. PROBLEMAS NA ÁREA CIENTÍFICA

A principal característica que atualmente diferencia o software científico dos demais está relacionada com o desenvolvimento do mesmo, que, geralmente, é feito pelos próprios especialistas, que possuem pouca familiaridade com técnicas de engenharia de software.

Esta característica faz com que a área científica enfrente vários problemas para a construção de software de qualidade. Este problemas podem ser explicados por diferentes motivos.

Em geral, os especialistas não estão preocupados em desenvolver um software produto com alto grau de manutenibilidade. Seus interesses estão voltados apenas para a descoberta de novas técnicas numéricas, validação de suas teorias e avaliação da "performance" através de testes cada vez mais complexos de seus algoritmos. [5]

Não há muita preocupação, por parte dos especialistas, em utilizar as técnicas de engenharia de software existentes e planejar as etapas de desenvolvimento, o que gera um ambiente de grande resistência à implantação de qualquer metodologia que fuja às suas atividades habituais.

Cross reforça esta argumentação quando afirma que as metodologias utilizadas no desenvolvimento de software numérico, na maioria das organizações, são informais. Elas, na maioria das vezes, se baseiam em um estilo próprio, que não pode ser facilmente explicado de maneira coerente. [4]

Quando o especialista implementa um algoritmo numérico visando sua validação, ele está utilizando, basicamente, conhecimento de sua área de atuação. Após esta validação, a fase de criatividade para os especialistas é considerada terminada. Desta forma, o produto de software inacabado (sem documentação, pouco estruturado, sem interface amigável, etc...) que foi gerado para tal validação, na maioria dos casos, torna-se o produto final de um projeto.

Os três motivos acima citados: a falta de interesse em gerar produtos com alto grau de manutenibilidade, a falta de estímulo para refinar o produto e a não utilização de ferramentas que auxiliem na modularização e padronização do software; contribuem para que o software científico, em geral, não tenha uma boa interface com o usuário, não possua uma documentação completa e adequada e seja pouco inteligível, sendo, portanto, pouco flexível e possuindo uma manutenção bastante difícil e onerosa.

O surgimento de ferramentas CASE não alterou este quadro. Principalmente porque ferramentas CASE automatizam técnicas já consolidadas na engenharia de software e, como já foi mencionado, não existe um padrão sobre métodos e técnicas eficazes para desenvolvimento de software científico. Alguns poucos desenvolvedores fazem uso de ferramentas um pouco mais poderosas (por exemplo, um depurador, um analisador de desempenho, ou ainda, analisadores estáticos e dinâmicos), mas, em geral, os especialistas desconhecem tais ferramentas. [9]

Desta forma, a grande maioria das empresas que possui desenvolvimento de software científico, se depara com problemas gravíssimos de qualidade de software. Um dos problemas mais sérios é o alto custo de manutenção dos programas, devido, principalmente, a falta de documentação durante o desenvolvimento que promove uma enorme dificuldade no entendimento do código.

A discussão, portanto, gira em torno de como o software científico pode ser desenvolvido pelo próprio especialista, utilizando técnicas da engenharia de software para estabelecer requisitos de qualidade, transformando este software científico em um produto com alto grau de qualidade, sem que a rotina habitual de trabalho do especialista seja muito alterada.

III. ENFOQUE PARA SOLUÇÃO

Cunto faz uma excelente constatação ao mencionar que os especialistas precisam, para aumentar a qualidade do software que desenvolvem e usam, de ferramentas que levem em conta tanto sua resistência a mudança quanto o uso de uma metodologia limitadamente informal. [5]

A criação de uma metodologia para desenvolvimento de software científico que considere o especialista no papel de usuário, colocando analistas e programadores como responsáveis pelo desenvolvimento, apesar de ser uma proposta bastante interessante, não conseguiria ser implantada com muita facilidade.

Primeiro, porque o software desenvolvido pelo especialista da área científica, de uma maneira geral, é extremamente dependente do conhecimento específico da área, tornando muito trabalhoso, custoso e não confiável, a transmissão de tal conhecimento do especialista para o analista ou programador. Segundo, porque a inclusão de profissionais de engenharia de software no processo de desenvolvimento aumentaria bastante o custo do software.

Desta forma, uma solução mais facilmente aceitável e implementável é a criação de um ambiente que torne viável o desenvolvimento e manutenção de software científico, com qualidade, pelo próprio especialista.

Este artigo apresenta a proposta de um ambiente CASE baseado em hipertexto que chamaremos de **HyFor**. O HyFor tem por objetivo controlar o desenvolvimento e manutenção de software científico desenvolvido em FORTRAN. A escolha do paradigma de hipertexto foi devido a sua adeqüabilidade ao problema e pela sua característica de fácil assimilação por usuários não especializados.

A principal característica do HyFor advém do fato do especialista ser induzido a utilizar o ambiente a partir de diversas facilidades que lhe são oferecidas, diminuindo o fator resistência a mudança. O ambiente também tenta se adaptar o máximo possível às atividades já rotineiras dos especialistas.

Os principais objetivos a serem alcançados com a utilização do HyFor é a diminuição do custo de manutenção do software e melhoria da qualidade do produto final desenvolvido.

A diminuição do custo de manutenção será obtida levando-se em conta que, para se realizar qualquer modificação em um programa, o programador gasta um tempo razoavelmente grande para, simplesmente, analisar e entender o código. Uma pesquisa recente mostrou que o estudo e análise do código consome em média 47 % do tempo total gasto na manutenção. [8]

Em ambientes científicos, onde não é objetivo principal dos especialistas produzir programas com alto grau de manutenibilidade e sendo a documentação extremamente escassa ou inexistente, o próprio especialista que desenvolveu o programa, após um certo tempo, gasta em média este mesmo percentual para entender seu próprio programa.

Sendo assim, pretendemos, através da utilização do enfoque de **hipertextos** no HyFor, permitir que um programa seja visualizado de forma não linear, através da navegação pelo código e por nós contendo informações que facilitarão o perfeito entendimento do programa, diminuindo, desta forma, o custo total de manutenção.

O ambiente proposto também permite a geração da documentação do programa, a partir dos nós do hipertexto, que sempre refletirá a realidade da versão em operação.

É fato que, tão ruim quanto não haver documentação, é existir uma documentação não atualizada sobre um programa. Desta forma, consideramos que a geração automática de uma documentação que seja atualizada e consistente, mesmo sendo simplificada, conseguirá aumentar a qualidade final de um novo programa gerado e também auxiliará sobremaneira o entendimento do mesmo na fase de manutenção, diminuindo, conseqüentemente, o custo dessa fase.

Embora não seja escopo desse trabalho resolver o problema de gerência de configuração [6], um primeiro passo é dado nesse sentido, pois o ambiente proposto também permite a criação de versões e um controle simplificado de alterações realizadas no código.

Em seguida são apresentadas as principais características do ambiente proposto.

IV. DESCRIÇÃO DO AMBIENTE PROPOSTO

O ambiente possui três ferramentas principais:

- Um Analisador de Código FORTRAN (ACF);
- Um Sistema Hipertexto;
- Um Gerador Automático de Documentação (GAD).

O ACF é responsável por executar todas as funções que dependem da análise estática do código, principalmente a geração automática dos nós e ligações do sistema hipertexto.

O sistema hipertexto permite a navegação pelos nós criados pelo ACF e a criação/alteração de alguns tipos de nós e suas ligações.

O GAD é responsável pela geração automática de uma documentação sobre o programa, a partir dos nós existentes no sistema hipertexto.

IV.1. ACF

O Avaliador de Código FORTRAN possui três módulos:

- Avaliador da complexidade do código;
- Avaliador da qualidade do código;
- Gerador (autoria) automático dos nós e ligações do hiperdocumento contendo informações sobre o código fonte, que será acessado (navegação) pelo sistema de hipertexto.

IV.1.1. Avaliador da Complexidade

Este módulo realiza a avaliação da complexidade do código através da análise estática de métricas de complexidade do código FORTRAN, visando gerar um relatório que contenha a quantificação de tais métricas.

O objetivo é utilizar estes resultados para caracterizar o ambiente de desenvolvimento através da obtenção de valores médios para as métricas, visando, posteriormente, a elaboração de estimativas, comparação entre projetos e controle da criação de código.

Este módulo se baseia quase que integralmente no trabalho desenvolvido por Bahia [2], onde as métricas utilizadas são descritas com mais detalhes.

Para melhor integrar a avaliação da complexidade do código, resultante da análise do ACF, com os demais módulos do ambiente, o relatório poderá ser considerado um nó do hiperdocumento. Este nó estará ligado ao nó do módulo principal, sendo as partes do código referenciadas no relatório geradas como botões que apontam para o programa.

A Figura 1 apresenta um exemplo do relatório de avaliação da complexidade gerado para um programa FORTRAN. Neste podemos identificar as métricas que são avaliadas pelo HyFor.

Relatório de Avaliação da Complexidade

Data: 19/04/93
Usuário: Carla

Módulo	A	B	C	D	E	F	G	H	I	J	L	M	N	O	P	Q	R	S	T	U	V	X
MESH	2	2	4	19	15	0	0	30	0	20	23	10	3	0	0	33	20	33	40	161	1	19
POTOBA	8	8	8	37	29	0	0	16	0	20	64	13	3	0	0	77	61	77	47	280	0	119
POLATE	8	8	12	55	43	0	0	16	0	20	70	14	3	0	0	84	67	84	56	309	0	138
BETPON	4	4	16	74	58	0	0	25	0	20	30	10	3	0	0	40	27	40	37	131	0	68
BETCON	4	4	20	93	73	0	0	25	0	20	30	10	3	0	0	40	27	40	38	132	0	68
COLCYL	16	16	24	115	91	0	0	15	0	20	105	19	3	0	0	124	102	124	55	466	0	279
BASCOL	16	16	28	136	108	0	0	14	0	20	107	18	3	0	0	125	104	125	54	466	0	276
Total	58	58	112	529	417	0	0	141	0	140	429	94	21	0	0	523	408	523	327	1945	1	967
Média	8	8	16	75	59	0	0	20	0	20	61	13	3	0	0	74	58	74	46	277	0	967

Legenda:

A - McCabe	B - McCabe Modificado
C - Vars. Gbls. não usadas	D - Vars. Gbls. que chegam
E - Vars. Gbls. usadas	F - Número de Goto's
G - Número de Con. Lógicos	H - Percentual de Comentários
I - Percentual de Brancos	J - Número de Vars. de E/S
L - Linhas de Código	M - Linhas de Comentário
N - Linhas de Decl. de Dados	O - Diretivas ao Compilador
P - Linhas em Branco	Q - Linhas Entregues
R - Linhas Executáveis	S - Total de Linhas
T - Operandos Distintos	U - Total de Operandos
V - Operadores Distintos	X - Total de Operadores

Figura 1 - Exemplo de um relatório de avaliação da complexidade gerado pelo HyFor

IV.1.2. Avaliador da Qualidade

Este módulo se baseia em um trabalho de pesquisa realizado pela Petrobrás, que resultou em um relatório chamado "Práticas Recomendadas para Programação FORTRAN" [7]. Neste são colocadas sugestões para a criação de programas FORTRAN de melhor qualidade. São abordadas as seguintes características: legibilidade, portabilidade, eficiência, modularidade e boa documentação.

A avaliação da qualidade corresponde ao resultado da análise feita pelo ACF sobre alguns pontos do relatório citado acima, considerados pelos próprios especialistas da Petrobrás, fundamentais à manutenibilidade e à eficiência do código. O ACF apenas informa os pontos que devem ser melhor avaliados, através de um relatório, o código não é alterado.

O relatório de avaliação da qualidade do código, resultante da análise do ACF, também poderá, se desejado, ser considerado um nó do hiperdocumento. Este nó estará ligado ao nó do módulo principal, sendo as partes do código referenciadas no relatório geradas como botões para o programa, da mesma forma como no Avaliador de Complexidade.

IV.1.3. Gerador do Hiperdocumento

A geração dos nós do hipertexto é feita em duas etapas:

1ª) Criação de arquivos contendo a marcação de cada tipo de nó, seus conteúdos e os botões de ligação entre eles, a partir da análise do código;

2ª) Geração, a partir destes arquivos, dos nós do hiperdocumento, levando em conta as características do sistema hipertexto utilizado.

Este módulo do ACF foi desenvolvido nestas duas etapas visando uma maior modularidade do sistema e uma melhor flexibilidade, pois permite que seja utilizado um outro sistema de hipertexto, alterando-se somente a segunda etapa do processo.

O Gerador do Hiperdocumento utiliza como entrada um programa FORTRAN e todas as subrotinas e funções que são chamadas por ele ou entre si, compilados sem erros. Chamaremos cada parte do programa, ou seja, o programa principal, as subrotinas e as funções, de módulos.

A função principal do gerador do hiperdocumento é criar a partir do conjunto de módulos do programa, o grafo dos módulos em hipertexto. Cada nó deste grafo será do tipo ***Módulo*** e conterá o código fonte de um módulo.

Além de montar este grafo, o gerador criará para cada nó do tipo Módulo, a partir da análise do código fonte, um conjunto de nós do tipo ***Operando***, um conjunto de nós do tipo ***Comentários*** e um nó do tipo ***Documentação***. Também é criado automaticamente um nó do tipo ***Observações***, vazio inicialmente (mais a frente veremos como gerar o conteúdo deste tipo de nó). Se existir algum operando do tipo arquivo, o sistema verificará se o arquivo está disponível na área do esquema do hiperdocumento e, se estiver, criará automaticamente, um nó do tipo ***Arquivo*** para cada arquivo existente.

Durante a geração do hiperdocumento, o gerador oferece ao usuário a opção de geração dos relatórios de avaliação da complexidade e de avaliação da qualidade. Caso estes sejam gerados, dois outros nós serão criados automaticamente: nó do tipo ***Avaliação de Complexidade*** e nó do tipo ***Avaliação de Qualidade***, contendo, cada um, seus respectivos relatórios. Estes nós serão ligados ao nó que contém o código fonte do programa principal e a todos os outros nós do tipo Módulo que são referenciados nos relatórios.

Cada determinado conjunto de nós fará parte de um contexto mais amplo. Os nós do tipo Módulo fazem parte do contexto ***Grafo do Código Fonte***. Os nós do tipo Operando e os nós do tipo Arquivo de cada nó do grafo, fazem parte do contexto ***Dicionário de Dados***, os nós Comentários de cada nó do grafo fazem parte do contexto ***Comentários Código Fonte***. O nó do tipo Documentação e o nó do tipo Observações são únicos para cada nó do grafo e não existe um contexto lógico para estes tipos de nós. (Figura 2)

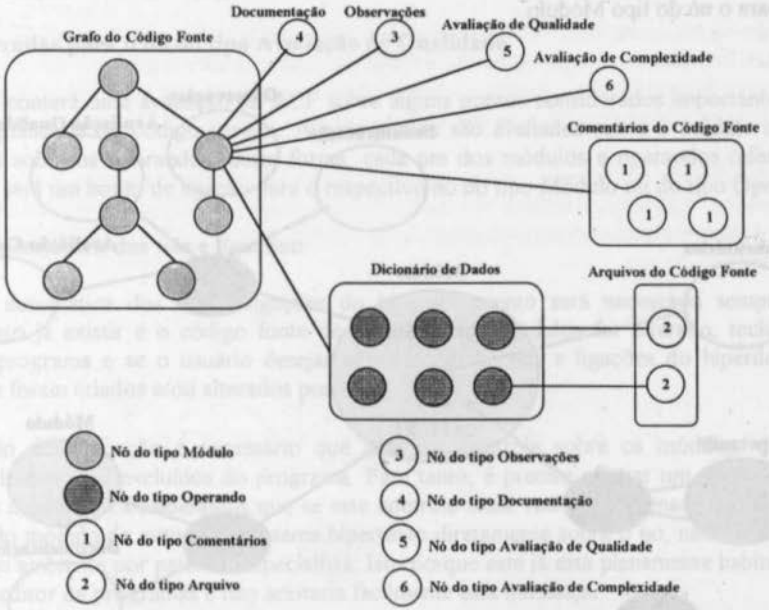

Figura 2 - Esquema simplificado dos tipos de nós, contextos e ligações do HyFor

O detalhamento do conteúdo de cada tipo de nó pode ser encontrado em [1].

O ACF gera, automaticamente, ligações referenciais, não tipadas e unidirecionais, entre os nós do hiperdocumento. A maioria das ligações têm como ponto de origem um nó do tipo Módulo.

• Ligações geradas para o nó do tipo Módulo

Cada nó do tipo Módulo estará ligado aos nós dos módulos que ele chama, montando o grafo de chamadas do programa. Será possível acessar, a partir do módulo que chama, o nó do tipo Módulo que contém o código fonte ou apenas o nó do tipo Documentação do módulo chamado.

Cada nó do tipo Módulo também estará ligado a todos os nós do tipo Operando, sendo que existirá um nó intermediário que direcionará a ligação para uma das duas partes que compõem o nó Operando. O botão desta ligação é o próprio nome do operando.

Cada nó do tipo Módulo será ligado, da mesma forma, ao nó do tipo Documentação, mesmo que não exista conteúdo para o nó. O mesmo ocorrendo com os nós do tipo Observações. O botão de ligação para estes dois tipos de nós será o mesmo, sendo utilizado, também, um nó intermediário para direcionamento da ligação.

Os nós do tipo Avaliação de Complexidade e Avaliação de Qualidade só serão ligados ao nó do tipo módulo que contém o código fonte do módulo principal do programa.

A Figura 3 mostra uma representação gráfica de todas as ligações geradas automaticamente pelo ACF, para o nó do tipo Módulo.

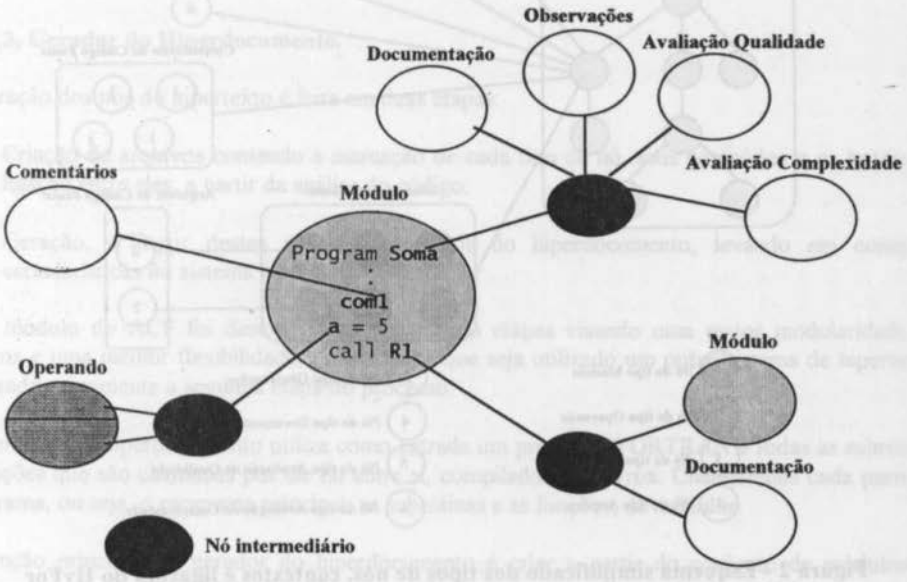

Figura 3 - Esquematização das ligações do nó do tipo Módulo

- **Ligações geradas para o nó do tipo Operando**

A parte que contém a descrição do operando, deste tipo de nó, não possui botões, mas a outra parte poderá, ou não, possuir botões de ligações com outros nós do tipo Operando. Isto ocorrerá quando um operando for uma variável global do tipo parâmetro, para a qual existirá um histórico que apresentará todo o caminho percorrido por este parâmetro entre os módulos (de onde vem e para

onde vai) e quais os operandos sinônimos utilizados. Os nomes dos operandos sinônimos serão botões para o nó intermediário do operando correspondente.

Quando um operando for do tipo Arquivo, poderá existir uma ligação com o nó do tipo Arquivo correspondente. Esta ligação só será feita automaticamente se o ACF conseguir criar o nó do tipo Arquivo também automaticamente. Entretanto, o usuário, se desejar, poderá criar manualmente esta ligação no módulo autoria do sistema hipertexto. O botão desta ligação será o nome externo do arquivo, contido no nó Operando.

• **Ligações geradas para o nó do tipo Avaliação de Complexidade**

Este relatório conterá informações quantitativas sobre métricas de complexidade para cada módulo que compõe o programa. O nome de cada módulo referenciado no relatório será um botão de ligação para o respectivo nó do tipo Módulo.

• **Ligações geradas para o nó do tipo Avaliação de Qualidade**

Este relatório conterá uma avaliação do ACF sobre alguns pontos considerados importantes para o aumento da qualidade do código gerado. Alguns pontos são avaliados sobre o módulo como um todo e outros sobre os operandos, desta forma, cada um dos módulos e operandos referenciados pelo relatório será um botão de ligação para o respectivo nó do tipo Módulo ou do tipo Operando.

Regeração Automática dos nós e ligações:

A regeração automática dos nós e ligações do hiperdocumento será necessária sempre que o hiperdocumento já existir e o código fonte de um ou vários módulos for alterado, incluído e/ou excluído do programa e se o usuário desejar não destruir os nós e ligações do hiperdocumento corrente, que foram criados e/ou alterados por ele.

Para execução desta função é necessário que haja um controle sobre os módulos que foram alterados, incluídos e/ou excluídos do programa. Para tanto, é preciso efetuar um controle sobre a edição destes módulos. Consideramos que se este controle fosse realizado apenas quando a edição fosse feita pelo módulo de autoria do sistema hipertexto diretamente sobre o nó, não tornaria viável a utilização do ambiente por parte do especialista. Isto porque este já está plenamente habituado com seu rotineiro editor de programas e não aceitaria facilmente esta mudança.

Desta forma, decidimos que seria melhor efetuar o controle através da verificação de alterações nos arquivos que contêm o código fonte de cada módulo, sem mexer no hiperdocumento. Assim o especialista poderia utilizar o editor de seu costume, sendo mais fácil a adaptação ao ambiente.

Seguindo este enfoque, sempre que o especialista entrar no hipertexto, será verificado se algum módulo foi alterado, incluído e/ou excluído. Se foi, será emitida uma mensagem para o especialista avisando que foram realizadas alterações e o hiperdocumento precisa ser regerado.

Esta função permitirá que se crie até cinco versões de um hiperdocumento e de cada módulo alterado. As versões dos hiperdocumentos serão independentes e serão acessadas em esquemas diferentes. As versões de cada módulo, quando criadas, farão parte do mesmo hiperdocumento.

Este procedimento foi adotado porque seria extremamente complicado e não confiável, manter versões de todos os tipos de nós em um mesmo hiperdocumento. Desta forma, decidimos permitir que o usuário tenha acesso a versão anterior do módulo alterado, no mesmo hiperdocumento e, caso deseje verificar a relação desta antiga versão com todo o programa, ele poderá navegar pelo hiperdocumento da versão anterior, em um outro esquema.

A função de Regeração do Hiperdocumento permitirá a criação de dois novos tipos de nós: tipo *Versão* e tipo *Alteração*.

O nó do tipo Alteração conterá a descrição das alterações realizadas no módulo em relação a versão anterior e estará ligado ao nó do tipo Módulo que contém a versão mais recente do código fonte. O nó do tipo Versão conterá a versão anterior do código fonte e também estará ligado ao nó do tipo módulo da mais recente.

O botão de ligação será o mesmo para estes dois tipos de nós, o nome do módulo que contém a versão mais recente do código, sendo necessário para esta ligação também a utilização de um nó intermediário. Será utilizado o mesmo nó intermediário para direcionar as ligações de um nó do tipo Módulo, para os nós do tipo Documentação, Observações, Alteração e Versão.

O nó do tipo Versão criado, por sua vez, somente possuirá ligações para os nós do tipo Comentário, Documentação, Alteração e Versão. Os dois últimos só existirão se houver uma versão ainda mais antiga. A Figura 4 apresenta uma esquematização das ligações do nó do tipo Módulo incluindo estes dois outros tipos de nós.

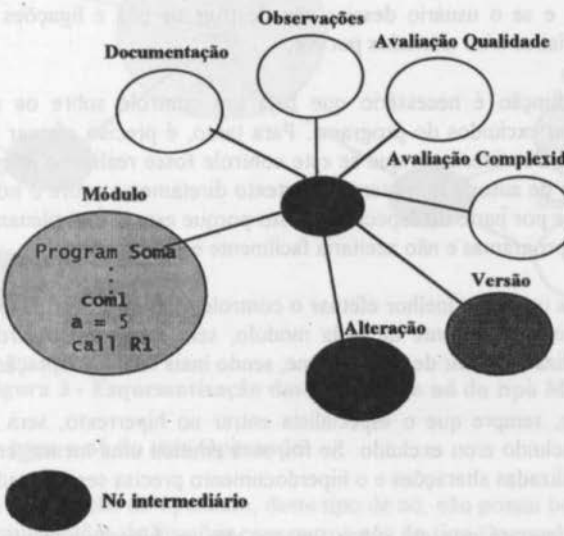

Figura 4 - Esquematização das ligações dos nós tipo Versão e Alteração com o nó do tipo Módulo

IV.2. Sistema Hipertexto

Como já foi mencionado, o ACF foi projetado para permitir que, através de pequenas alterações, seja possível a geração/regeração automática do hiperdocumento para qualquer sistema hipertexto. O ambiente considera a utilização de qualquer sistema hipertexto que possua as seguintes características principais:

- Possibilidade de criação de nós tipados e ligações referenciais, viabilizando a criação da rede semântica projetada para o HyFor;
- Controle de acesso ao sistema visando a segurança, controle e integridade dos hiperdocumentos;
- Módulos de autoria e navegação do sistema bem independentes para que seja possível o controle dos nós que não poderão ser alterados;
- Interface bastante amigável, para que o especialista tenha facilidade na utilização da ferramenta;

O protótipo apresentado neste artigo utilizou o sistema hipertexto ***Hiperbase*** [3] e o módulo de geração/regeração de hiperdocumento do ACF foi desenvolvido levando em conta as características básicas deste sistema.

IV.3. GAD

Esta ferramenta é responsável pela geração automática de documentação sobre o programa, na forma de relatórios. Todos os documentos gerados são obtidos a partir dos nós do hiperdocumento do esquema corrente no ambiente.

Quatro tipos de documentos podem ser gerados pelo GAD:

- Especificação do programa;
- Especificação do módulo;
- Relatório de controle de alterações do programa;
- Relatório de controle de alterações de um módulo;

V. DESCRIÇÃO DO PROTÓTIPO

O protótipo do HyFor foi implementado para microcomputadores com processador 80386 em diante, utilizando o sistema operacional DOS, e precisa de 4 megabytes de memória disponível. A exigência do tamanho de memória é do Analisador de Código FORTRAN (ACF).

Este ambiente foi escolhido devido a grande utilização em áreas científicas, atualmente, de software em linguagem FORTRAN para microcomputadores e de existir uma grande perspectiva de crescimento desta área.

V.1. Implementação do ACF

O ACF, de uma maneira geral, é um analisador estático de código FORTRAN segundo a sintaxe definida para o produto MICROSOFT FORTRAN. Produto que foi desenvolvido de acordo com as seguintes normas: American National Standard Programming Language FORTRAN, ANSI X3.9-

1978 (também conhecido como FORTRAN 77) e a International Organization for Standardization, ISO 1539-1980 Programming Languages - FORTRAN.

O ACF foi desenvolvido em linguagem C e funciona da seguinte maneira: um módulo que chamamos de corpo principal recebe como entrada o programa FORTRAN e o analisa guardando dados em forma de tabelas que serão utilizados como entrada para cada um dos módulos que compõem o ACF. Cada um desses módulos foi implementado de acordo com a especificação descrita no capítulo anterior.

Na Figura 5 apresentamos uma visão esquemática dos módulos do ACF e seus interrelacionamentos.

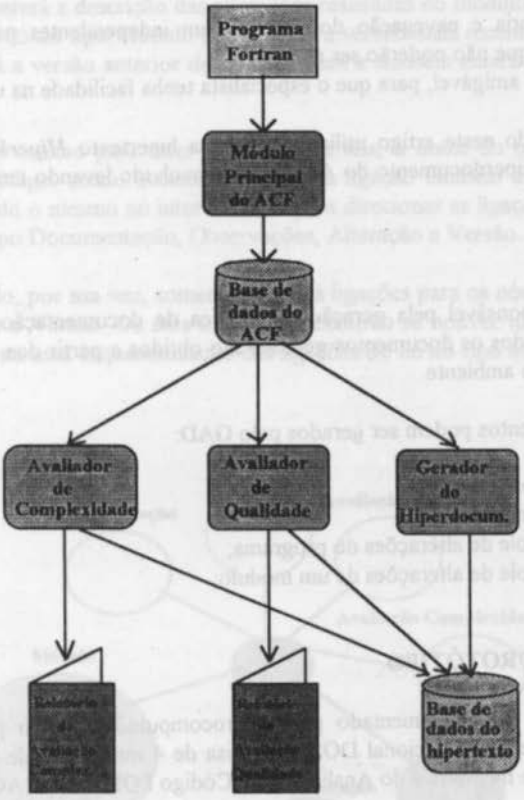

Figura 5 - Visão esquemática do ACF

V.2. Implementação do Sistema Hipertexto

O Sistema Hipertexto integrado ao ambiente proposto é o **HIPERBASE/DOS** que está sendo desenvolvido pela equipe do projeto Hiperbase da Área de Desenvolvimento do NCE/UFRJ [3]. O estágio avançado de desenvolvimento deste projeto permitiu que este fosse utilizado no protótipo do HyFor.

O Hiperbase está dividido em três módulos: módulo de definição, módulo de autoria e módulo de navegação. O módulo de definição serve para definir o esquema conceitual no qual a base de dados do hiperdocumento será construída. O módulo de autoria permite a criação dos nós e ligações do hiperdocumento. O módulo de navegação permite a navegação por estes nós através de suas ligações.

No HyFor utilizamos somente os módulos de autoria e navegação porque a definição do esquema conceitual da base de dados é feita fora do Hiperbase, em uma função independente do ambiente.

O Hiperbase suporta além das características de um sistema hipertexto, características de um sistema gerenciador de banco de dados, sendo possível definir descritores para um nó, que serão manipulados por uma linguagem de manipulação de dados.

Esta facilidade do Hiperbase também não é utilizada no HyFor porque os nós gerados automaticamente pelo ACF não possuem descritores.

V.3. Implementação do GAD

O GAD foi desenvolvido em linguagem C e é um gerador automático de documentação que possui seis alternativas de geração de documentos: Especificação do Programa, Especificação do Módulo, Relatório de Controle de Alterações do Programa, Relatório de Controle de Alterações de um Módulo, Relatório de Avaliação da Complexidade e Relatório de Avaliação da Qualidade.

Os quatro primeiros documentos são gerados a partir dos nós do hiperdocumento, os dois últimos são resultado dos módulos do ACF de avaliação da complexidade e de avaliação da qualidade, respectivamente. O conteúdo de cada documento e os tipos de nós dos quais cada informação é extraída, foram especificados no capítulo anterior.

VI - CONCLUSÃO

Novas culturas são difíceis de se introduzir. Mudanças no método de trabalho devem ser implantadas com planejamento e com condições favoráveis a sua consolidação. Este trabalho descreve uma ferramenta/ambiente voltada para implantar técnicas de engenharia de software em ambientes científicos a partir da solução de problemas mais imediatos e que afetam mais visivelmente os produtores de software desses ambientes, ou seja a manutenção de programas.

Considerando que a maioria desses programas são mal documentados, construídos a partir de especificações não explícitas, procurou-se criar um ambiente que facilitasse a inclusão gradativa de documentação e especificação, pelo próprio especialista, com um interface amigável e simples. Espera-se assim reduzir o tempo e conseqüentemente o custo, do trabalho de manutenção desses programas.

O enfoque de hipertextos, conforme esperado, se mostrou bastante adequado à organização do programa em módulos interligados. Embora o sistema de hipertexto utilizado tenha apresentado problemas, pois ainda se encontra em desenvolvimento, a implementação do ambiente foi bastante

facilitada por ele. É relativamente fácil a sua troca por outro sistema de hipertexto, já que apenas um dos módulos do ambiente precisará ser trocado.

O ambiente carece ainda de testes mais rigorosos para o seu aprimoramento e correção de eventuais problemas. Os testes realizados até o momento, embora com resultados promissores, precisará ser ampliado para garantir a sua disseminação com sucesso. O objetivo do trabalho que visava demonstrar a praticidade da proposta foi atingido.

A evolução do trabalho prevê, além de testes mais rigorosos e em maior número, a melhoria do sistema de hipertexto ou a troca por um outro sistema, o refinamento dos relatórios de avaliação de qualidade e complexidade, e a geração do Diagrama de Estrutura Modular do programa. Esta última é de vital importância para dar seqüência a idéia de implantação de técnicas de engenharia de software de forma "bottom-up".

Além disso, um aprimoramento do sistema de controle de versões permitirá que uma efetiva gerência de configurações seja realizada. A evolução nesse sentido faz parte da estratégia inicialmente descrita de utilização gradativa de técnicas de engenharia de software em ambientes de desenvolvimento de software científico.

Referências Bibliográficas

[1] Alves, C.G., "Ambiente de Apoio ao Desenvolvimento e Manutenção de Software Científico", Dissertação de Tese de Mestrado, COPPE/UFRJ, abril/1993.

[2] Bahia, A.S., "O Uso de Métricas de Complexidade para o Controle da Qualidade de Software Científico", Dissertação de Tese de Mestrado, COPPE/UFRJ, maio/1992.

[3] Borges, M.R.S., "Suporte de Bancos de Dados para Sistemas Hipertextos", XVIII Conferência Latinoamericana de Informática, Ilhas Canárias, Espanha, Agosto/1992.

[4] Cross, M., Moscardini, A.O. e Lewis, B. A., "Software Engineering Methodologies for Scientific and Engineering Computation", Appl. Math. Modelling, vol.10, outubro/1986.

[5] Cunto, W., Araujo, J., Giovannetti, F e Rivero, J., "FPLUS: Programming Environment for Scientific Applications", IEEE Software, setembro/1991.

[6] Narayanaswamy, K. e Scacchi, W., "Maintaining Configurations of Evolving Software Systems", IEEE Transactions on Software Engineering, vol. SE-13, num. 3, Março/1987.

[7] PETROBRÁS/DEPEX, "Recomendações para Programação FORTRAN", novembro/1989.

[8] Swanson, E.B., "The Dimension of Maintenance", Proc. 2nd Int. Conf. Software Engineering, IEEE, pp.492-497, 1976.

[9] Werner, C.M.L., "Reutilização de Software no Desenvolvimento de Software Científico", dissertação de tese de doutorado, COPPE/UFRJ, março/1992.