AMBIENTE DE APOIO AO DESENVOLVIMENTO E MANUTENÇÃO DE SOFTWARE CIENTÍFICO

Carla Gama Alves

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Marcos Roberto da Silva Borges, PhD.
(Presidente)

Ana Regina Cavalcanti da Rocha, D.Sc.

Fernando Silva Pereira Manso, PhD.

RIO DE JANEIRO, RJ - BRASIL
ABRIL 1993

ALVES, CARLA GAMA

Ambiente de apoio ao desenvolvimento e manutenção de software científico. [*Rio de Janeiro*] 1993,

XII, 152p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1993)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Controle de qualidade de software científico 2. Hipertexto

I. COPPE/UFRJ II. Título (série)

À minha mãe Nádia cujo carinho e dedicação sempre foram meus maiores sustentos.

Ao Paulo que, com seu amor e companheirismo, foi o mais importante apoio que eu poderia ter para conseguir finalizar este trabalho.

AGRADECIMENTOS

Aos meus irmãos Adriano, Fernanda e Flávia que são meus melhores amigos e incentivadores e ao Itamar que, desde a sua chegada, me ensina muito com sua grande experiência de vida.

Ao meu pai Valter que foi um dos grandes responsáveis por toda a minha formação.

À toda minha família (incluindo agregados) que com sua união se torna o melhor apoio que alguém pode precisar.

À Maria que foi uma linda descoberta neste período.

Ao Nelson Ebecken que com sua grande experiência foi uma peça chave em um dos momentos mais importantes de todo esse processo.

Ao meu super amigo Einstein que sempre me incentivou e me ensinou muito com seu jeito todo próprio de ser.

Ao Heitor que esteve presente desde o início de minha vida profissional, sempre me estimulando e apoiando.

Ao Álvaro Bahia pela sua imensa ajuda e estímulo no início do desenvolvimento do protótipo e pelas dicas de implementação do ACF.

Ao Alessandro Cerqueira ("Jacaré") e Sérgio Prallon pela grande ajuda na fase de programação do HyFor.

Ao Márcio pela elaboração da tela de entrada e ao Luiz e Fernanda pelos trabalhos realizados no Hiperbase.

Aos colegas do NCE que sempre me ajudaram com muita presteza e com os quais compartilhei toda minha apreensão durante a elaboração desta tese.

Aos colegas do CENPES que me ajudaram a levar a frente este trabalho, estando sempre disponíveis para me atender e esclarecer dúvidas. Em especial ao Levy que permitiu a utilização do programa Mesh no exemplo de utilização.

Aos colegas do EDISE e CEPEL pela ajuda na pesquisa inicial realizada para esta tese.

A todos os meus amigos, que através de seu interesse e estímulo muito me incentivaram a não interromper este trabalho. Quero que saibam que cada um, em particular, é responsável por um pedaçinho desta tese.

Finalmente, ao meu orientador pela paciência e incentivo ao longo deste trabalho. Todo esse processo foi uma grande experiência para mim e ele foi um grande exemplo de profissional, amigo e pessoa. À você, Marcos, toda a minha admiração e amizade.

Foi difícil mas valeu a pena.

À todos MUITO OBRIGADO.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências (M.Sc.).

AMBIENTE DE APOIO AO DESENVOLVIMENTO E MANUTENÇÃO DE SOFTWARE CIENTÍFICO

Carla Gama Alves
Março de 1993

Orientador: Marcos Roberto da Silva Borges

Programa : Engenharia de Sistemas e Computação

Os problemas que atualmente envolvem as atividades de desenvolvimento e manutenção de software científico, promovem um sensível aumento no custo deste software durante seu ciclo de vida.

É preciso transformar estas atividades em um processo controlado e previsível, inserindo métodos, técnicas e ferramentas que promovam melhoria de qualidade no software sem, entretanto, prejudicar os critérios de qualidade estabelecidos pelos especialistas. Para tanto, é proposto um ambiente de apoio ao desenvolvimento e manutenção de software científico que, através da utilização do conceito de hipertexto, permitirá a criação de software com maior qualidade e tornará a fase de manutenção menos onerosa.

Chamamos o ambiente proposto de HyFor e este é composto por ferramentas que juntas permitirão atingir os objetivos descritos acima.

As principais ferramentas são: um Analisador de Código Fortran (ACF) que gera automaticamente um hiperdocumento a partir de um programa Fortran, gera um relatório de avaliação de qualidade e outro de avaliação da complexidade e reestrutura o código; um Sistema Hipertexto que é utilizado para navegar e alterar o hiperdocumento gerado; e um Gerador Automático de Documentação (GAD) que permite a geração automática de documentos a partir do hiperdocumento.

Abstract of the Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M.Sc.).

Carla Gama Alves
March, 1993

Thesis Supervisor: Marcos Roberto da Silva Borges

Department: Computing and Systems Engineering

Nowadays, the problems that concern the scientific software development and maintenance activities, promote a sharp increase on the software costs during its life cycle.

It is necessary to transform these activities into a controlled and predictable process, inserting methods, techniques and tools that can improve software quality without harming the quality criteria established by the experts. For that purpose, a supporting environment to the development and maintenance of scientific software is proposed. This environment makes use of hypertext concept, aiding the creation of software with higher quality, turning the maintenance phase less costly.

The proposed environment is called HyFor and it is composed by a set of tools in order to achieve the goals described above.

The main tools are: a Fortran Code Analyser (ACF) which generates automatically an hyperdocument from a Fortran program, a report of quality evaluation and other of complexity evaluation and the code restructuring; a Hypertext System which is used to browse and alter the generated hypertext; and a Documentation Automatic Generator (GAD) which allow the automatic generation of documents from the hyperdocument.

Índice

Lista de Figuras

I - Introdução

O capítulo I apresenta uma introdução ao trabalho de tese, descrevendo sua motivação, objetivos e organização do documento.

I.1. Motivação

A principal motivação para realização deste trabalho foi verificar que a área científica é extremamente dependente da informática e enfrenta sérios problemas no desenvolvimento de seus programas, os quais promovem um sensível aumento no custo total do software científico.

Isto se deve, principalmente, ao fato deste software ser desenvolvido pelo próprio especialista da área, o qual está somente interessado em validar e tornar eficiente seus algoritmos numéricos, não se importando em gerar um produto que atenda à critérios de qualidade pré-estabelecidos.

Os especialistas não utilizam, efetivamente, nenhum método ou ferramenta de engenharia de software para desenvolvimento de seus programas, apesar de todos terem plena noção da importância e necessidade de tal utilização. Todos começam o desenvolvimento pela codificação e acabam no teste que, por sinal, também não é realizado com base em algum método específico. Os especialistas reclamam a falta de ferramentas automatizadas para auxiliar o desenvolvimento do software, alegando que sem estas é praticamente impossível o desenvolvimento do software científico dentro de métodos e técnicas da engenharia de software.

Nenhum documento de especificação é gerado. Por vezes, um ou outro especialista gera algum tipo de anotação particular sobre o programa. Quando o programa é utilizado não só pelo responsável pelo desenvolvimento, é gerado um pequeno Manual do Usuário, mas, do contrário, nenhum documento é gerado.

Os especialistas possuem dificuldade de realizar manutenção nos programas por eles próprios desenvolvidos. Eles usam uma forma de desenvolvimento muito particular e com o tempo acabam se esquecendo de detalhes adotados e, como já foi dito, quase sempre, não existe qualquer tipo de documentação. Por este motivo também é extremamente difícil a utilização de outros técnicos ou especialistas para realização da tarefa de manutenção, tendo em vista que o tempo dispendido no entendimento do programa será muito alto.

Um especialista bem qualificado, cujo custo por hora é bastante alto, quando tem um programa em produção, gasta a maior parte de seu tempo em tarefas de manutenção, ao invés de estar utilizando sua capacidade intelectual no desenvolvimento de outros programas.

Existe também bastante dificuldade na manutenção de programas científicos comprados de outras empresas. Estes programas sempre são obtidos juntamente com o código fonte e sempre são bastante adaptados pelos especialistas. A documentação existente também é muito precária e gasta-se muito tempo na análise do código fonte para entendimento do programa.

Por estes motivos, podemos considerar que os ambientes de desenvolvimento de software científico enfrentam atualmente uma séria "crise de software", onde o desenvolvimento não possui controle de qualidade, normas e padrões, sendo realizado de forma extremamente artesanal e pessoal, transformando sua manutenção em uma atividade difícil e bastante onerosa.

Sendo assim, a visão deste problema, tão sério e custoso às empresas que desenvolvem software científico, foi um grande estímulo para tentar buscar dentro dos métodos e técnicas disponíveis na engenharia de software, uma possível proposta de solução amenizadora

I.2. Objetivo

Esta tese tem como objetivo propôr um ambiente de apoio ao desenvolvimento e, principalmente, a manutenção de software em Fortran, visando o aumento da produtividade em áreas científicas. Este ambiente tentará captar o máximo possível de informações sobre o software sem, entretanto, exigir que o especialista altere sobremaneira sua rotina habitual de trabalho.

Este ambiente utilizará o conceito de hipertexto visando um melhor e mais fácil entendimento do programa durante a fase de manutenção, objetivando diminuir o custo total desta fase.

As características que envolvem o conceito de hipertexto deverão permitir também que sejam gerados, automaticamente, alguns relatórios que documentem o programa, aumentando a qualidade do produto final gerado.

A conclusão sobre a maneira de solucionar o problema descrito nesta tese será

atingida a partir de um estudo da literatura sobre as características e principais problemas enfrentados no desenvolvimento de software científico e um estudo feito também sobre a técnica de hipertextos, onde já foram constatadas vantagens de sua utilização em algumas áreas de aplicação, possibilitando imaginar esta técnica servindo de apoio ao desenvolvimento e manutenção de software científico, com obtenção de resultados bastante satisfatórios.

I.3. Organização da tese

Esta tese está dividida em sete capítulos.

O capítulo I define os objetivos e motivação para este trabalho e apresenta a organização deste documento.

Os capítulos II, III e IV apresentam as dissertações das pesquisas realizadas na literatura que levaram a proposta desta tese.

O capítulo II descreve o que vem a ser qualidade de software, apresenta um método de como controlar e aferir esta qualidade, a falta de qualidade repercutindo na fase de manutenção e a necessidade de uma boa documentação visando amenizar tais problemas.

O capítulo III descreve o ambiente científico do ponto de vista da literatura, com abordagens encontradas na prática, caracterizando-o e apresentando suas principais diferenças com outros domínios de aplicação e os problemas específicos de seu desenvolvimento.

O capítulo IV descreve as principais características, vantagens e desvantagens de utilização e exemplos de aplicações em hipertextos.

O capítulo V apresenta a especificação informal do ambiente proposto nesta tese, que chamamos de HyFor.

O capítulo VI descreve o protótipo desenvolvido para o HyFor e apresenta um exemplo de utilização do ambiente.

O último capítulo corresponde a um resumo do que foi feito, uma descrição das conclusões do trabalho e sugestões para futuras pesquisas.

O apêndice A apresenta uma relação das métricas de complexidade encontradas na literatura e as utilizadas neste trabalho. O apêndice B apresenta um exemplo do Relatório de Avaliação da Complexidade e o apêndice C um exemplo do Relatório de Avaliação da Qualidade gerados pelo ACF. O apêndice D apresenta um exemplo dos relatórios gerados pelo GAD. O apêndice E apresenta a listagem do código fonte do programa MESH. O apêndice F apresenta a regra de padronização dos nomes dos nós gerados pelo ACF.

II- Aspectos de qualidade de Software

O capítulo II faz uma explanação sobre qualidade de software, descreve a importância da documentação do software e apresenta os principais problemas enfrentados pela fase de manutenção.

II.1. Controle de qualidade do software

Todos os problemas que envolveram a chamada "crise do software", no final dos anos 70, possuíam suas origens em um único ponto: falta de qualidade do software gerado.

A engenharia de software surgiu como uma nova disciplina apresentando métodos, técnicas e ferramentas com o principal objetivo, senão único, de produzir software com qualidade.

Existem diversas definições para qualidade de software. A definição dada por Rocha é a seguinte: "Um conjunto de propriedades a serem satisfeitas em determinado grau, de modo que o software satisfaça as necessidades de seus usuários." [ROCH87C]

Independentemente da definição estabelecida, é consenso que esta qualidade deve ser avaliada e controlada desde o início do desenvolvimento do software e que o grupo de desenvolvedores deve atuar consciente e efetivamente para atingi-la.

Para tanto, existem diversos métodos descritos na literatura que possibilitam esta avaliação e controle. Vale ressaltar que um método deve possuir as seguintes propriedades: [ROCH87C]

- Confiabilidade - isto é, as medidas obtidas utilizando o método são indicativas da qualidade do produto

- Efetividade - isto é, a avaliação pode ser realizada e contribui para um melhor entendimento do produto.

Neste trabalho apresentaremos, resumidamente, o método para avaliação da qualidade de software proposto por Rocha [ROCH83]. Este método baseia-se nos seguintes conceitos:

- Objetivos de qualidade - que são propriedades gerais que o produto deve possuir;

- Fatores de qualidade do produto - que determina a qualidade do ponto de vista dos diferentes usuários do produto (usuário final, mantenedores, etc...);

- Critérios - que são atributos primitivos possíveis de serem avaliados;

- Processos de avaliação - que determinam o processo e os instrumentos a serem usados de forma a se medir o grau de presença, no produto, de um determinado atributo;

- Medidas - que são resultado da avaliação do produto segundo os critérios;

- Medidas agregadas - que são o resultado da agregação das medidas obtidas ao avaliar segundo os critérios e quantificam os fatores.

A figura 2.1 apresenta a estrutura deste método.

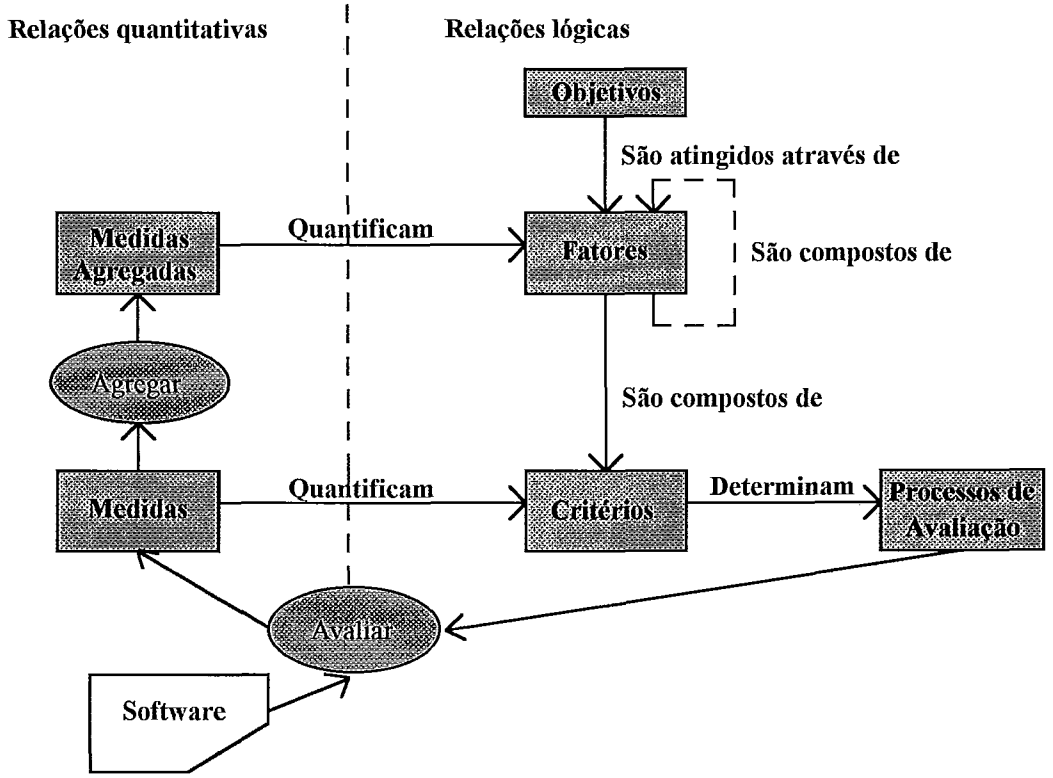


Figura 2.1. Estrutura do método para avaliação da qualidade de software

Este método possui as características de confiabilidade e efetividade e pode ser aplicado nas fases de especificação, projeto e codificação de programas. Para cada uma dessas fases, são definidos seus objetivos de qualidade, assim como, os fatores e critérios através dos quais os objetivos são atingidos. Os critérios são quantificados pelas métricas (medidas).

Neste trabalho consideraremos as métricas associadas à avaliação de programas. A figura 2.2 apresenta a descrição dos objetivos, fatores (sub-fatores) e critérios para a fase de codificação dos programas.

- Confiabilidade Conceitual
  - # Fidedignidade
    - * Completeza
    - * Necessidade
    - * Precisão
  - # Integridade
    - * Robustez
    - * Segurança
- Confiabilidade da Representação
  - # Legibilidade
    - * Clareza
      - + Número de decisões + Padronização
    - * Concisão
      - + Não anomalia + Não repetição
    - * Estilo de programação
      - + Comentário + Identificação + Indentação
      - + Organização visual + Programação estruturada
    - * Modularidade
      - + Balanceamento + Coesão + Não acoplamento
      - + Não memorização + Número de módulos inferiores
      - + Tamanho + Número de módulos superiores
  - # Manipulabilidade
    - * Disponibilidade
      - + Acessibilidade + Atualização
    - * Rastreabilidade
      - + Localizabilidade interna + Localizabilidade Externa
- Utilizabilidade
  - # Avaliabilidade
    - * Validabilidade
    - * Verificabilidade
  - # Eficiência
  - # Manutenibilidade
  - # Operacionalidade
    - * Amenidade ao uso
    - * Oportunidade
  - # Portabilidade
  - # Rentabilidade
  - # Reutilizabilidade

Figura 2.2. Critérios para a fase de codificação de programas [ANDR91]

II.2. Manutenção de software

II.2.1. A fase de manutenção

O processo de desenvolvimento de software contém três fases genéricas independentemente do paradigma de engenharia de software escolhido para o desenvolvimento: definição, desenvolvimento e manutenção [PRES87]. A figura 2.3 ilustra esta visão.

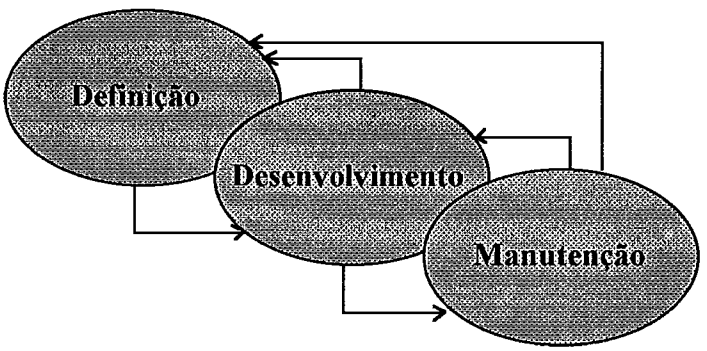


Figura 2.3. Paradigmas: uma visão genérica [PRES87]

Sendo a última fase do processo de desenvolvimento, as atividades desenvolvidas na fase de manutenção englobam todo trabalho realizado sobre um software após a sua implantação em ambiente operacional, até a sua morte.

Desta forma, a manutenção de software inclui diversas atividades desde a correção de erros que por ventura não tenham sido encontrados durante a fase de testes, até a adaptação do software a novas tecnologias, passando pelas tarefas de treinamento de novos grupos de usuários e respostas a dúvidas sobre a operacionalidade do sistema.

Vale ressaltar que não existe um consenso sobre a melhor definição para o termo ***manutenção de software***. Alguns acham que o termo deve ser utilizado da mesma forma como o é para sistemas físicos, ou seja, quando eles estão deteriorados [PARI83]. Algumas organizações usam termos tais como "melhoria" ou "redesenvolvimento de sistemas" para expressar atividades que classificamos aqui como manutenção de software.

A definição mais ampla deste termo é justificada pela abordagem gerencial, métodos e ferramentas de suporte que são similares mas diferem daquelas utilizadas no desenvolvimento inicial [BENN91]. Esta interpretação é consistente com a definição da IEEE [IEEE83].

II.2.2. Classificação de manutenção de software

Existem na literatura diversas classificações para as atividades que envolvem manutenção de software. Sendo que a mais amplamente utilizada considera quatro tipos: manutenção corretiva, manutenção adaptativa, manutenção aperfeiçoadora e manutenção preventiva. Estes termos foram estabelecidos por Swanson [SWAN76].

Manutenção corretiva ocorre quando são corrigidos erros no programa, que não foram identificados durante a fase de testes. Esta atividade engloba também o trabalho de atualização de documentação do software.

Manutenção adaptativa é necessária quando o software precisa se adaptar as novas tecnologias (hardware e software) implantadas no ambiente operacional.

Manutenção Aperfeiçoadora é realizada quando o software deve englobar novos requisitos do usuário ou realçar e trazer melhorias para maior valorização do software.

Manutenção preventiva ocorre quando o software é alterado para aumentar sua manutenibilidade ou confiabilidade ou para prover ao software melhor base para futuras alterações [PRES87]. Este tipo de manutenção é relativamente rara em ambientes de desenvolvimento.

Manutenção preventiva tem por objetivo investir na redução dos custos com os demais tipos de manutenção, melhorando a qualidade dos programas [CAVA92].

A figura 2.4 apresenta um quadro contendo as principais características de qualidade que um software deve apresentar segundo Ghezzi et al. [GHEZ91]

Um programa	é aquele
Correto	que comporta-se de acordo com a especificação
Confiável	do qual o usuário pode depender
Robusto	que se comporta de forma razoável, até em circunstâncias imprevistas
Eficiente	que usa os recursos computacionais de forma econômica
Amigável	cujos usuários consideram fácil de usar
Verificável	cujas demais qualidades podem ser facilmente verificadas
Manutenível	que permite alterações quando já em operação, tanto para corrigir defeitos, quanto para evoluir
Reusável	que pode ser utilizado na construção de outro
Portátil	que pode ser executado em diferentes ambientes
Compreensível	que possui comportamento previsível pelo usuário
Interoperacional	que possui a habilidade de coexistir e cooperar com outros programas
Produtivo	cujo processo de produção é eficiente
Oportuno	que é entregue ao usuário dentro dos prazos estabelecidos
Visível	cujo desenvolvimento está documentado claramente

Figura 2.4. Principais qualidades que um software deve apresentar [GHEZ91]

A figura 2.5 apresenta o percentual em média utilizado em cada tipo de manutenção.

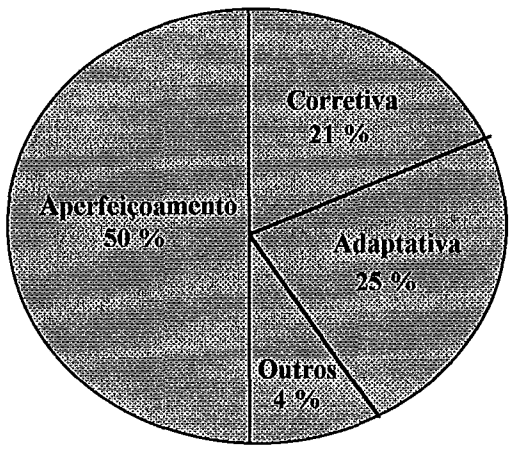


Figura 2.5. Distribuição percentual das atividades de manutenção [PRES87]

Outras classificações da fase de manutenção dividem, por exemplo, a manutenção de aperfeiçoamento em manutenção de melhoria e de eficiência. A primeira ocorre quando há alterações nos requisitos do usuário e a segunda aumenta a eficiência e legibilidade dos programas. [GORL91]

II.2.3. Custo de manutenção do software

Até recentemente, a fase de manutenção tinha sido bastante negligenciada pela literatura em relação as fases de definição e desenvolvimento do processo de desenvolvimento de software. Poucas pesquisas foram realizadas e abordagens técnicas ou métodos foram propostos. [PRES87]

O custo cada vez mais crescente desta etapa e a constatação de que é impossível produzir software, de qualquer tamanho, imune a necessidade de introdução de alterações, mudou este panorama. A preocupação atual é tornar o software suscetível a manutenção, desde a sua construção, visando a redução dos custos futuros de manutenção e, conseqüentemente, dos custos operacionais.

Software, entretanto, não se torna suscetível a manutenção acidentalmente. Para tanto, a manutenibilidade deverá ser perseguida desde o início do desenvolvimento. Software é suscetível a manutenção à medida que, desde a sua concepção, tenha sido especificado com tal propósito. [ROCH83] [OLIV85]

A atividade de manutenção de software em ambiente operacional é responsável por mais da metade do esforço total gasto no ciclo de vida deste software.

Pressman considera que este esforço está em torno de 60% [PRES87]. Fairley afirma que cerca de 40% a 60% do tempo de vida de um software são dedicados à esta atividade, podendo, em certos casos, chegar a 90%. [FAIR85] [GORL91]

Uma pesquisa realizada por Lientz [LIEN81], abrangendo 487 empresas, revelou que 50% do esforço total de desenvolvimento eram dedicados à manutenção, sendo 55% deste esforço voltados para a extensão de funcionalidade ou melhoramentos no desempenho, 25% para adaptar-se à mudanças nos dados ou no ambiente de processamento, e apenas 20% para correção de erros. Foi constatado também que cerca de 0,5 homem/ano era dedicado à manutenção de um sistema típico.

Vale ressaltar que ao longo do tempo os sistemas se tornam mais difíceis de serem mantidos porque também se tornam crescentemente mais complexos. [BAHI92]

Moreton publicou os resultados de uma pesquisa mostrando que, na época, o intervalo de esforço de manutenção era entre 5,3% e 90% do custo do ciclo de vida, enquanto dois anos antes era entre 10% e 70%. Ainda assim, a maioria das 26 empresas participantes, colocavam a fase de manutenção em segundo plano em relação ao

desenvolvimento. [MORE88]

Pressman vislumbra a "manutenção-bound" onde organizações podem deixar de produzir novos softwares por estarem todos os recursos disponíveis envolvidos com manutenção de softwares antigos.

A figura 2.6. mostra o percentual de orçamento total de um software gasto em manutenção, nas últimas duas décadas e na atual projetada.

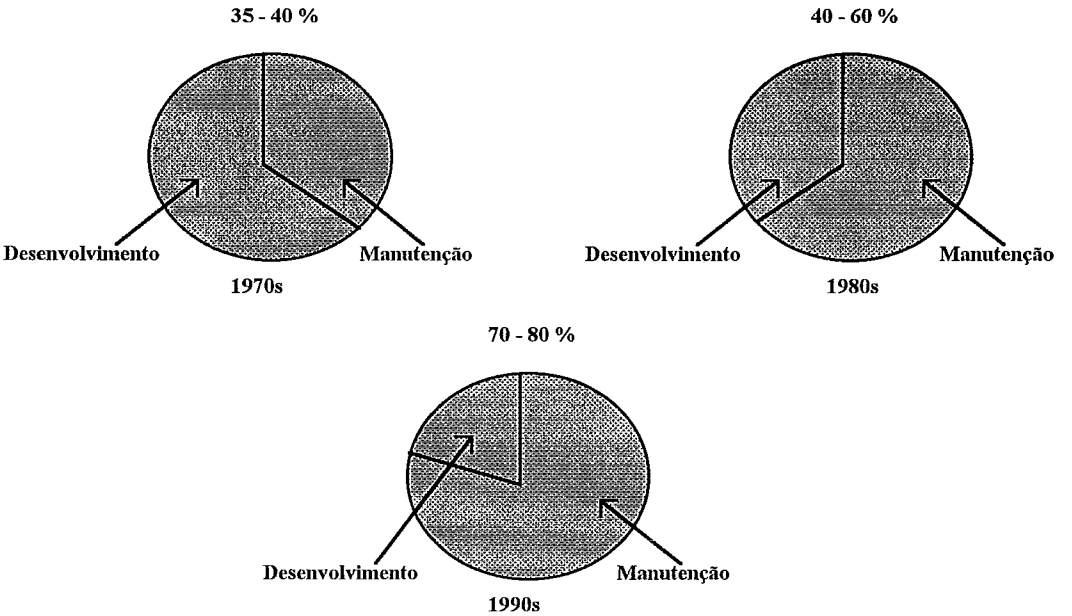


Figura 2.6. Percentual do custo de manutenção [PRES87]

II.2.4. Fatores que afetam o custo de manutenção do software

É incontestável a importância e o esforço envolvido na fase de manutenção de software e, tendo em vista que nenhum software é imune a manutenção, é extremamente necessário um melhor entendimento dos problemas que envolvem esta atividade e suas causas.

Uma das razões principais para o alto custo de manutenção de software é a crescente complexidade deste por causa do grande número de modificações feitas durante sua vida operacional. [GORL91]

Canning caracterizou manutenção de software como um "iceberg": Nós esperamos que o que está imediatamente visível é tudo que existe. [CANN72]

Na realidade, sabemos que uma enorme quantidade de problemas e custos em

potencial residem sob a superfície.

Este problema é extremamente agravado quando a versão inicial do software foi mal estruturada, possuindo um nível muito baixo de qualidade.

Programas ilegíveis exigem um grande esforço para que um programador possa entendê-lo.

A falta de modularidade ajuda a propagação do "efeito cascata" promovido por alterações no código. Este é um problema muito grave e ocorre quando erros são infiltrados no programa quando uma alteração é realizada e não são detectados pelos testes, os quais, na maioria das vezes, se detêm a parte do código alterada.

A inexistência de uma documentação adequada e atualizada pode induzir a erros de interpretação e conseqüentemente problemas nas alterações efetuadas. Ter um boa documentação é uma importante ajuda não somente na manutenção atual, mas também para melhorar a manutenibilidade de programas e diminuir os custos de manutenções futuras. Vale ressaltar que uma documentação é tão ruim quanto não ter uma documentação, pois pode levar o programador a entender o programa incorretamente, ou pode fazê-lo levar mais tempo para compreender a lógica do código.

Outros fatores também contribuem fortemente para o alto custo de manutenção, são eles: [GORL91]

- falta de entendimento dos requisitos do usuário;
- falta de treinamento adequado do uso de sistema;
- atribuição das tarefas de manutenção a programadores inexperientes; e
- pouco tempo estabelecido pelas empresas para tarefas de manutenção.

II.2.5. Soluções propostas para os problemas de manutenção

Todos os problemas descritos anteriormente podem, em parte, serem atribuídos ao grande número de programas existentes atualmente que foram desenvolvidos sem nenhum embasamento da engenharia de software. Uma metodologia disciplinada não deve ser vista como uma panaceia. Entretanto, engenharia de software provê pelo menos soluções parciais para cada problema associado com manutenção [PRES87]

Podemos considerar, inclusive, que a manutenibilidade é o objetivo chave que guia os passos de uma metodologia de engenharia de software. Manutenibilidade pode ser definida qualitativamente como a facilidade com a qual o software pode ser entendido, corrigido, adaptado e/ou melhorado.

Algumas técnicas para reduzir o esforço na manutenção de software são descritas abaixo: [GORL91]

- Estratégias para visualização do código na tela que auxiliem o entendimento do programa;
- Facilidades de documentação automática do programa que gere documentação atualizada do código em operação;
- Analisadores inteligentes de código;
- Reestruturação do software e re-engenharia;
- Engenharia reversa;
- Abordagem de programação defensiva;
- Uso de boas ferramentas de desenvolvimento;
- Uso de boas estratégias de projeto.

As três primeiras técnicas citadas acima são utilizadas na manutenção imediata do software e se preocupam essencialmente em melhorar seu entendimento, tendo em vista que é esperado gastar-se 47 % do tempo total de manutenção apenas na análise do código. As outras são técnicas que visam reduzir custos de manutenções futuras.

II.2.6. Considerações gerais

Não somente aspectos técnicos são importantes para garantir a diminuição do custo total de manutenção, aspectos gerenciais para organização desta fase também possuem um grande peso. A manutenção de software precisa da combinação destes dois fatores.

Documentar bem as alterações realizadas no código também é uma tarefa essencial para o controle e gerenciamento da manutenção. É muito importante ter

disponível o histórico de alterações contendo a descrição do problema, as alternativas avaliadas, problemas encontrados e solução adotada, bem como as antigas versões do programa. Estas informações podem diminuir sobremaneira o tempo gasto na análise estratégica para solução do problema.

II.3. Documentação de software

II.3.1. Importância

"Documentar consiste em descrever leis fundamentais da psicologia da forma para construir documentos mais claros e fáceis de serem lidos. A não utilização destas leis pode fazer com que o número de erros e interpretações incorretas cresça dramaticamente" [BUEN89].

É indiscutível a necessidade e importância de documentar sistemas de computador, visando uma melhor produtividade dos profissionais de sistemas em geral e uma diminuição do custo total do software.

A grande maioria das empresas/organizações atuantes na área de informática, certamente, já sofreram ou estão sofrendo as consequências da falta de normas e padrões para documentação de seus softwares. Mas até hoje ainda nos deparamos com questões básicas relativas a documentação de software: [KRON87]

- Como fazer com que os profissionais da área documentem seus softwares ?
- Como obter qualidade em documentação ?
- Como fazer com que a documentação produzida seja efetivamente utilizada nas áreas usuárias dos produtos, como fazer com que os usuários realmente leiam a documentação que lhes é entregue ?

Estas questões são consequências da própria falta de entendimento/ conscientização que ainda existe por parte dos profissionais de informática. Na maioria dos casos, este profissional considera a documentação como a última etapa do desenvolvimento de um sistema, tornando esta tarefa extremamente desmotivante e não atrativa, levando em conta que ao final de um projeto geralmente os profissionais já estão envolvidos em outros projetos [KRON87].

O fato também da documentação do software, na maioria dos ambientes de desenvolvimento, ser vista, infelizmente, como uma tarefa não criativa, faz com que esta atividade seja considerada pouco nobre e, conseqüentemente, de pouca importância.

O problema pode residir no fato de que os profissionais da área consideram documentação de software apenas o conjunto de documentos necessários a utilização e a manutenção do software, ou seja, manuais dos usuários e manuais do sistema.

A documentação de software precisa ser vista como uma consequência natural de todo o processo de desenvolvimento. Inclusive, uma certa documentação existe intuitivamente para cada participante de um projeto. Desde o levantamento dos dados, onde ele registra de alguma forma tais informações, até a codificação dos programas, ele passa por uma documentação mental das atividades. O que falta são uma sistemática, normas e padrões que o obrigue a materializar tais documentos e torná-los parte do sistema. Assim como um engenheiro não levanta uma ponte sem ter todo o projeto documentado em forma de desenhos, cálculos e padrões; um analista de sistemas não deveria construir um sistema sem ter feito uma análise detalhada das necessidades do usuário, apresentada como um documento de especificação; e um projeto minucioso sobre como implementar a nova sistemática proposta, apresentado como um documento de projeto.

Sendo adotada uma sistemática de desenvolvimento, os profissionais da área enxergarão naturalmente a documentação como uma parte integrante de todo o processo de desenvolvimento de software.

Algumas empresas, a maioria de grande porte, já adotaram esta filosofia de desenvolvimento e geram seus softwares dentro de regras e padrões de documentação pré-estabelecidos. Entretanto algumas destas empresas ainda pecam quando exigem um conjunto de documentos mas também exigem um cronograma extremamente apertado para elaboração dos mesmos, prejudicando, sobremaneira, a qualidade final destes documentos. Com esta mentalidade, o documento de baixa qualidade freqüentemente é deixado de lado após a implantação do software, fugindo aos objetivos de sua geração.

Desta forma, vale ressaltar que, não basta ter normas e padrões de documentação, é necessário estabelecer o nível de qualidade desejado para cada documento e criar procedimentos para controle desta qualidade.

Para resolver o problema de qualidade da documentação gerada durante o processo de desenvolvimento de software, diversas regras e critérios já foram estabelecidas na literatura. [ROCH87C] [PRES87] [FAIR85] [YOUR85] [BEIZ84] [DEUT82].

Todos esses autores tentam atingir objetivos de qualidade de acordo com premissas básicas para geração de uma documentação eficiente: [BUEN89]

- atender as necessidades básicas pretendidas, sendo eficaz;
- estar adaptada à realidade;
- ser o mais simples possível, precisa e confiável;
- ser perfeitamente entendida por quem a utiliza ;
- apresentar facilidade de manutenção;
- sensibilizar a seus usuários, quanto a importância de sua utilização e quanto aos benefícios que proporciona.

Grice ratifica a importância da geração de documentação de software, considerando que "os produtos de computador, hoje em dia, consistem de três componentes: hardware (equipamentos físicos de um computador e periféricos), software (os programas que rodam no hardware), e informação (as direções para o entendimento e uso do hardware e software)" [GRIC88]. E considera, por isso, que o esforço de desenvolvimento de um produto requer o envolvimento de três grupos de pessoas: os responsáveis pelo desenvolvimento de hardware, os responsáveis pelo desenvolvimento de software e os responsáveis pelo desenvolvimento da informação. Estes três grupos de profissionais devem seguir processos de desenvolvimento similares, paralelos, para produzir produtos integrados e utilizáveis.

II.3.2. Documentos de um software

Podemos considerar como fazendo parte da documentação de um software todo documento que auxilie o desenvolvimento, a implantação, a operação e a manutenção do software, ou seja, todas as informações necessárias para promover a comunicação entre usuários finais e todos os profissionais envolvidos no processo de desenvolvimento e manutenção do software.

Desta forma, a documentação de um software inclui: [MAID92]

- documentos que descrevem os produtos gerados durante seu ciclo de vida;
- comentários no código fonte;
- manuais de utilização do sistema;
- informações "on-line".

II.3.2.1. Documentos do ciclo de vida

Como já foi visto no item II.2 deste capítulo, o processo de desenvolvimento de software contém três fases genéricas: definição, desenvolvimento e manutenção.

Na fase de definição, a preocupação é definir "o que" precisa ser feito para resolver o problema em questão. Genericamente podemos considerar como produto desta primeira fase, um documento que contenha todo o planejamento de desenvolvimento, objetivos e os requisitos detalhados sobre o que o usuário deseja.

Podemos encontrar na literatura diversos roteiros para documentos, gerados nesta fase de definição. [ROCH87] [PRES87] [FAIR85] [BUEN89] [DEMA79].

Levando em conta que o produto final gerado nesta fase é um documento ou conjunto de documentos, e que tais documentos serão a base para as próximas fases, podemos enfatizar a grande importância em efetuar um rígido controle de qualidade sobre esta documentação.

Na fase de desenvolvimento, o objetivo principal a ser atingido é definir "como" o software especificado na fase anterior deverá ser desenvolvido e desenvolvê-lo, atingindo os requisitos do usuário. Documentos desta fase incluem especificações que traduzem os requisitos em um conjunto de representações computacionais, a documentação de todos os programas e relatórios de testes.

Também encontramos na literatura, roteiros para documentos gerados na fase de desenvolvimento. [PRES87] [FAIR85] [ROCH87] [BUEN89]

Nesta fase temos como produtos gerados: especificações, relatórios, códigos e manuais. Assim como na fase anterior, também é fundamental para o sucesso do software, um efetivo controle de qualidade sobre os documentos gerados.

A fase de manutenção prevê documentos que controlem alterações efetuadas no software, que são associadas a correções de erros, adaptações necessárias ao ambiente ou melhorias requisitadas pelo usuário.

II.3.2.2. Comentários no código fonte

Na etapa de codificação, o trabalho consiste em traduzir um algoritmo para uma determinada linguagem de programação. Cada linguagem possui uma sintaxe própria e determinadas regras que devem ser seguidas. Cada programador pode utilizar tais regras, ou a própria sintaxe, de forma diferente.

Os manuais dos programas são o produto final da documentação dos programas, e são essenciais, mas nestes são descritas informações genéricas sobre o programa como, objetivo, variáveis utilizadas, rotinas que chama, etc.... Para um perfeito entendimento do

código também se faz necessário descrever o fluxo de controle utilizado, identificar trechos do código e relacioná-los com trechos do projeto, etc.... Este tipo de documentação se faz sob a forma de comentários no código fonte.

Desta forma, levando em conta, principalmente, que geralmente programas são manipulados por diversas pessoas durante seu desenvolvimento e manutenção, os comentários servem como comunicação entre estes profissionais, promovendo um perfeito entendimento sobre o código fonte.

Vale ressaltar que, mesmo informações contidas nos manuais de programas, podem facilitar e agilizar a manutenção dos programas se estiverem descritas no próprio código, em forma de comentários. Tais informações são consideradas constituintes do cabeçalho do programa. Na literatura podemos encontrar alguns exemplos de cabeçalhos de programas. [MAID92] [PRES87] [FAIR85].

Um cuidado que deve ser tomado em relação a comentários no código fonte é o de não sobrecarregar demais o código dificultando a análise fluente dos comandos. [YOUR75]

II.3.2.3 Manuais de utilização do software

A parte o fato do software atingir os requisitos do usuário e de seu código fonte ser bem construído e documentado, existe a necessidade de gerar uma documentação que auxilie o usuário na utilização do produto.

Esta documentação inclui: Manual do Usuário, Guia do Operador e Manual de Referência.

- Manual do Usuário: objetiva orientar o usuário na utilização do software. Usa uma terminologia adequada com o ambiente para o qual o software foi desenvolvido;

- Guia do Operador: objetiva orientar o pessoal de suporte, descrevendo o ambiente operacional do software;

- Manual de Referência: normalmente é utilizado por usuários com alguma experiência no mesmo tipo de aplicação. Contém informações básicas sobre o software.

II.3.2.4 Informações "on-line"

Com o advento de recursos de software e hardware cada vez mais sofisticados, é possível considerar mais um grupo de informações como fazendo parte da documentação de um software, são as informações "ON-LINE".

Brown define documentação ON-LINE como "comunicação projetada para ser apresentada em telas de terminal de vídeo de forma a facilitar interações entre software de computador e os indivíduos que o gerenciam, o operam ou o mantêm." [BROW86 em SHIR88]

Existem algumas taxonomias para informações ON-LINE, os próprios autores consideram tal classificação bastante flexível. A figura 2.7 apresenta uma taxonomia de documentação ON-LINE [SHIR88].

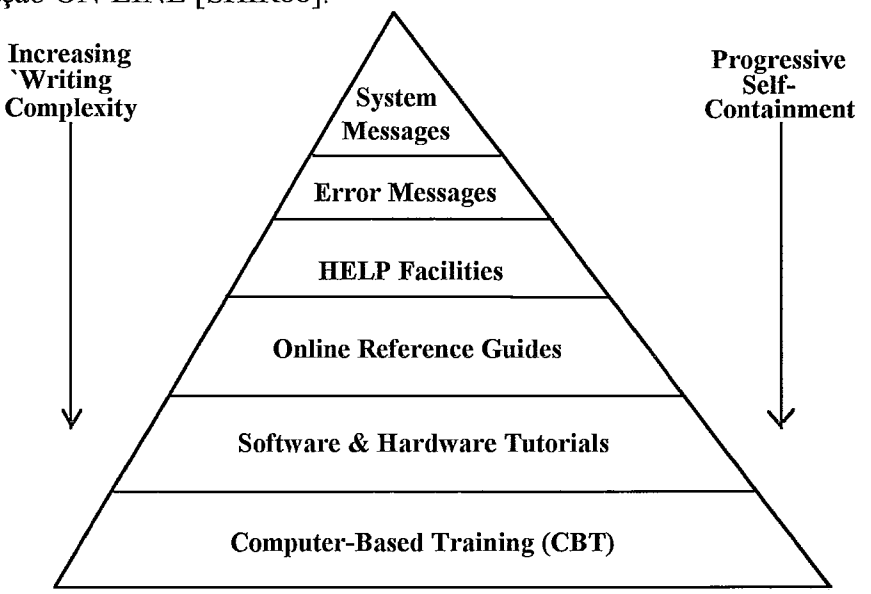


Figura 2.7 - Taxonomia de Documentação ON-LINE

- Mensagens do Sistema: informação, instruções e indicações de problemas, que podem aparecer durante a operação de produtos de software;

- Mensagens de Erro: informações que orientam o usuário na identificação de erros;

- Facilidades de Ajuda (HELP): informações que direcionam o usuário, dentro do sistema de software, para solucionar problemas.

- Guias de Referência ON-LINE: informações detalhadas, compreensíveis, sobre software ou hardware que são estruturadas como dicionários ou enciclopédias e acessadas através de diversos esquemas de indexação;

- Tutoriais de Software e Hardware: Conjunto de instruções ON-LINE consistindo de informações procedurais passo-a-passo ilustradas com exemplos e gráfico;

- Treinamento Baseado em Computador: Inclui todas as formas de uso de computadores no suporte ao aprendizado. Isto pode incluir outros recursos ("media") tais como, vídeo, fitas de áudio, etc....

Este tipo de documentação não é considerada como substituta da tradicional documentação impressa. Entre uma e outra existem muitas diferenças, vantagens e desvantagens. Por isso devemos enxergar as informações ON-LINE como uma forma de documentação diferente, adicional e complementar, visando tornar o software mais consistente e de uso mais amigável.

A utilização de interfaces gráficas na área de informações ON-LINE é uma recente novidade, e traz uma grande vantagem, que é a de substituir palavras por imagens (ícones), tornando a interface independente de idiomas. Esta tecnologia tem a capacidade de descrever com simplicidade operações detalhadas e/ou complexas. [MAID92]

II.3.3. Automatização da documentação do software

Como já foi visto, os produtos gerados em cada uma das fases de criação de um software, são basicamente documentos e código fonte. Cada um contendo representações textuais e gráficas de acordo com o método, técnica e ferramenta de engenharia de software utilizados em cada fase.

Por isso, podemos considerar como ferramentas para automatização da documentação do software, não só os editores e processadores de texto, formatadores e similares, mas também diversas ferramentas e ambientes integrados que automatizam atividades ou fases do ciclo de vida do software.

II.3.3.1. Histórico

O processo de conscientização da necessidade de estabelecer paradigmas para o desenvolvimento de software que culminou na disciplina de Engenharia de Software, trouxe imediatamente à tona, a necessidade de automatização das ferramentas propostas por esta disciplina, visando uma melhor produtividade.

A criação de ferramentas individuais, ou seja, ferramentas que dão suporte a uma atividade específica dentro do ciclo de vida do software, não facilitou sobremaneira a utilização da engenharia de software, devido ao fato das atividades serem extremamente dependentes umas das outras e necessitarem de um grande controle na transferência de informações entre elas.

Atualmente existe uma concentração de esforços no sentido de criar ambientes integrados automatizando todo o ciclo de vida do software. Um ambiente consiste de várias ferramentas diferentes, juntamente com um mecanismo para integração destas ferramentas.

É inegável que, apesar de suas limitações, a disponibilidade destas ferramentas está ajudando, lentamente, a deslocar o foco de atenção dos desenvolvedores, de detalhes do código para o contexto completo de desenvolvimento [SMIT90], promovendo a desejada conscientização dos profissionais para geração da documentação necessária durante as fases do ciclo de vida de um software.

II.3.3.2. Algumas ferramentas existentes

Como exemplos de ferramentas que auxiliam o processo de elaboração de documentos, temos os processadores de texto **MicroSoft Word** e o **Chi-Writer**, o formatador de textos **XDVI- previewer** e o editor dirigido a sintaxe,**VAX Language-Sensitive Editor (VAXLSE)** [MAID92].

As ferramentas CASE, **Cradle** e **JSP Workbench** são implementações dos métodos análise, projeto e programação estruturada. **Time Bench**, **Card Tools** e **Prosa** são ferramentas específicas para desenvolvimento de sistemas de tempo real e **Excelerator**, **Adagen** e **Smart System**, são exemplos de ferramentas CASE de propósito geral [SMIT90]. O **SIM** integra as atividades de codificação e elaboração da documentação do código em um mesmo ambiente [MAID92].

Como exemplos de ambientes CASE integrados temos o **EAST IPSE** que é um protótipo que provê um ambiente integrado para desenvolvimento de aplicações durante todas as fases do ciclo de vida, para workstations Unix. O ambiente **Epos** que suporta diferentes métodos de desenvolvimento [SMIT90]. O **DIF** que é um sistema de hipertexto que ajuda a integração e o gerenciamento dos documentos produzidos e usados durante o ciclo de vida de um software [GARG89]. E o **DynamicDesign** que é um ambiente CASE para linguagem C, ele armazena código fonte C, requisitos e outros documentos, em uma base de dados hipertexto [BIGE87].

II.4. Conclusão preliminar

Este capítulo mostrou os aspectos gerais de qualidade de software, dando ênfase aos tópicos de qualidade atacados neste trabalho de tese.

O primeiro item apresentou um método para avaliação e controle da qualidade e enfatizou a utilização deste método na fase de codificação, onde os programas são avaliados segundo critérios estabelecidos. O ambiente proposto nesta tese possui uma ferramenta que analisa um código Fortran quantificando métricas de complexidade que servirão para uma posterior avaliação qualitativa do código através de comparações e regressões estatísticas. E outra ferramenta que avalia aspectos de qualidade gerais, levando em conta critérios considerados importantes para o ambiente científico em particular.

O segundo item apresentou os problemas enfrentados pela fase de manutenção de software, estabelecendo bases para validação da proposta estratégica utilizada neste trabalho, que considera a visualização do código Fortran utilizando o enfoque de hipertexto e da necessidade de uma documentação sempre atualizada do código e de suas alterações.

O terceiro item descreveu a atividade de documentação de software, enfatizando sua fundamental importância na geração de software de qualidade.

O próximo capítulo apresentará o ambiente científico, descrevendo suas características e problemas enfrentados em obter produtos de software de qualidade. Apontando seus principais problemas, os quais residem, principalmente, na falta de documentação e no alto custo da fase de manutenção.

III - Software Científico

O capítulo III descreve o ambiente científico, caracterizando-o e apontando os maiores problemas enfrentados no desenvolvimento de software nesta área.

III.1. Caracterização

O software científico, ou software numérico, se diferencia dos demais por diversos fatores, incluindo desde a sua estrutura física até a seqüência de passos utilizada no seu desenvolvimento.

Este tipo de software surge da necessidade de se validar algoritmos numéricos que têm como função expressar a solução encontrada para problemas de modelagem físico-matemática dos fenômenos pesquisados. Estes algoritmos possuem um alto grau de complexidade, com manipulação de uma grande quantidade de dados e execução de muitos cálculos. [ARAU89]

O tipo de processamento de software científico é considerado, geralmente, como "CPU-BOUND", ou seja, é caracterizado por uma grande quantidade de cálculos, requerendo muito espaço em memória principal e grande utilização de CPU. [MAID88]

Werner contesta esta consideração e argumenta que, a partir de sua experiência com o desenvolvimento de software científico, pôde perceber o aparecimento, cada vez maior, deste tipo de software com características muito mais de "I/O-BOUND", ou seja, software que utiliza intensivamente dispositivos de E/S para manipulação de dados. E atribui esta transformação a diversos fatores, dentre os quais, a rapidez de processamento dos atuais equipamentos, o aumento de tamanho e complexidade das atuais aplicações, até a proliferação de máquinas e ferramentas que exigem controles mais específicos. [WERN92]

Em relação a linguagem de programação utilizada no desenvolvimento deste tipo de software, podemos considerar o Fortran um consenso. É fato que outras linguagens também são utilizadas no ambiente científico, tais como, ADA, PL/I, C e Pascal, mas o Fortran é, sem dúvida, a linguagem que, efetivamente, caracteriza o software científico.

Podemos, inclusive, considerar que o Fortran ainda permanecerá bastante tempo em evidência, levando em conta um fator fundamental: a existência de grandes bibliotecas de programas científicos/matemáticos de escopo internacional nesta linguagem, devido ao

Fortran ser a primeira linguagem de alto nível a considerar aspectos de hardware relevantes a execução de cálculos científicos, fazendo com que diversos programas fossem desenvolvidos e amplamente divulgados em revistas e jornais científicos. [RICE71]

Segundo Arsac, "Uma linguagem de programação passa a definir uma comunidade de pessoas que a compreende" [ARSA84], desta forma, a existência destas bibliotecas de âmbito internacional, tornou o Fortran a linguagem oficial de comunicação no meio científico.

Sua utilização foi cada vez mais crescente devido ao fato de ocorrer, neste meio, uma efetiva reutilização de código e, sendo este escrito em Fortran, a tendência natural foi, e é até hoje, desenvolver os novos programas também em Fortran.

Apesar das críticas existentes contra o Fortran, é fato que os próprios especialistas não se queixam de suas limitações porque esta linguagem os atende perfeitamente nos aspectos que lhes interessam. Alguns inclusive comentam que não pretendem deixar de programar em Fortran por considerar esta a melhor linguagem existente para processamento matemático.

Desta forma, podemos considerar que o Fortran ainda tem uma longa vida pela frente, principalmente com a nova versão do Fortran 90 que tentou rever uma série de problemas encontrados nas versões anteriores (Fortran IV, Fortran 77 e Fortran 8x) e adicionou uma série de novas facilidades. [WERN92]

Dentre as áreas de abrangência do desenvolvimento de software científico, será dada ênfase, neste trabalho, à área de engenharia.

III.2. Problemas no desenvolvimento

A característica principal que atualmente diferencia o desenvolvimento do software científico dos demais, se encontra no fato de que, geralmente, os próprios especialistas desenvolvem o software numérico que eles precisam e, quase sempre, estes especialistas não possuem experiência, também, em ciência da computação. Desta forma, o grau de qualidade que o software precisa possuir em sua versão final, nem sempre é atingido.

E apesar de, em geral, utilizarem equipamentos bastante sofisticados, as únicas ferramentas utilizadas por estes especialistas que desenvolvem suas próprias aplicações, se resume em editores e compiladores. [CUNT91]

O fato do software científico ser desenvolvido pelos próprios especialistas, sem qualquer tipo de especificação antes da codificação, já é uma característica antiga, desde os primórdios da programação. É claro que neste início não havia tanta preocupação com o projeto do software. A evolução deste fator se deu, primeiramente, para o desenvolvimento de outros tipos de aplicações. Para o caso do software científico a importância dada à etapa de projeto do software é considerada recente. Esta preocupação surgiu exatamente do fato da produção do software científico começar a enfrentar problemas na obtenção de produtos de qualidade em tempo hábil. [BELL86]

Entretanto, devido ao alto grau de complexidade e especialidade que os algoritmos numéricos envolvem, é possível entender o motivo pelo qual o próprio especialista precisa desenvolver o software que irá validar este algoritmo e a dificuldade de se alterar esta realidade.

III.3. Aspectos de qualidade

Os problemas enfrentados na obtenção de produtos de software de qualidade, em termos de engenharia de software, na área científica, podem ser explicados por diferentes motivos.

O principal deles está no fato dos especialistas não estarem preocupados em desenvolver um software produto com alto grau de manutenibilidade. Eles têm seus interesses voltados apenas para a descoberta de novas técnicas numéricas, validação de suas teorias e avaliação da "performance" através do teste de problemas cada vez mais complexos. Eles não se preocupam em conhecer as técnicas de engenharia de software existentes e planejar as etapas de desenvolvimento do software.

Para os especialistas não treinados em ciência da computação, solucionar um problema científico é muito mais importante do que aprender novas metodologias de desenvolvimento ou linguagens de programação mais avançadas [CUNT91]. Na verdade, mesmo que o especialista passe mais da metade do seu tempo com serviços de desenvolvimento do software, ele, ainda assim, não acha importante o seu aprimoramento nesta área.

Outro motivo que agrava os problemas de qualidade enfrentados pelo software científico é que, quando se tenta validar um algoritmo numérico através da sua implementação, muito esforço é dispendido e a fase de criatividade para os especialistas, após esta validação, é considerada terminada. Desta forma, o produto inacabado que foi gerado para tal validação, na maioria dos casos, torna-se produto final.

Um terceiro motivo que explica a falta de qualidade do software científico é o fato de que o software desenvolvido pelo próprio especialista, geralmente, possui características extremamente pessoais, ou seja, o software é desenvolvido de acordo com o bom senso, sem utilização de nenhum método que poderia vir a estruturar e padronizar o desenvolvimento dos programas.

Cross ratifica este ponto de vista quando ele chega a afirmar que a metodologia utilizada no desenvolvimento de software numérico, na maioria das organizações, é informal. Elas muitas vezes se baseiam em um estilo próprio, que não pode ser facilmente explicado de maneira coerente. [CROS86]

Esta situação é ainda mais prejudicada pela falta de reconhecimento que existe com os programadores que implementam um excelente software produto sem, entretanto, promover alguma inovação ao respectivo algoritmo que gerou o software. Os especialistas consideram sem criatividade e o trabalho não é tão valorizado quanto deveria ser. [CROS86]

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 não tenha uma boa interface com o usuário, não possua uma documentação completa e adequada e seja pouco inteligível, sendo, portanto, totalmente inflexível e quase manutenível, levando-se em conta que a manutenção se torna muito difícil e onerosa. Conseqüentemente, não atingindo os requisitos de qualidade essenciais a qualquer tipo de software.

Uma boa interface com o usuário é uma exigência da época que vivemos, devido ao advento das facilidades de geração de interfaces em microcomputador. Em passado recente, uma entrada de dados era normalmente feita na própria linguagem de desenvolvimento do software científico, que é na maior parte dos casos a linguagem Fortran. Atualmente o usuário está mais conhecedor das técnicas existentes e por isso se torna mais exigente. Desta forma uma boa interface com o usuário é um requisito de qualidade importantíssimo para o produto de software.

Uma documentação completa é muito importante tanto para estabelecer a comunicação entre desenvolvedores e usuários, possibilitando a utilização do produto, como, principalmente, para facilitar a manutenção do mesmo, visto que este tem como uma de suas característica, vida útil longa e, por isso, necessidade de contínuas manutenções.

A modularização e padronização do desenvolvimento dos programas auxilia a inteligibilidade do programa pelas pessoas que irão realizar manutenções futuras e, como já foi mencionado, o software científico necessita de contínuas manutenções sendo, desta forma, necessário que o produto contenha um alto grau de manutenibilidade.

III.4. Métodos e técnicas

Todos os problemas enfrentados pelo desenvolvimento de software científico ainda são mais agravados pelo fato de não existir um consenso sobre a forma mais adequada para elaboração deste software e pela dificuldade, ainda existente, em conceituar e medir certos aspectos referentes ao grau de qualidade que deve ser atribuído a este.

É fato que qualquer software precisa ser utilizável, confiável, eficiente, de uso amigável e de fácil manutenção. Mas a determinação do peso que cada um destes fatores deve exercer sobre a composição final do objetivo de qualidade sofre influências devido: ao ambiente operacional, ao ambiente de desenvolvimento, à visão do usuário e a fatores circunstanciais. [BAHI88]

A discussão gira em torno de como o software científico pode ser desenvolvido para atingir os objetivos do engenheiro, com a participação do profissional de computação, utilizando as técnicas da engenharia de software, para estabelecer requisitos de qualidade, transformando este software científico em um produto com alto grau de qualidade.

Existem algumas propostas de metodologias para desenvolvimento: [BAHI88], [CROS86], [DODD82], [PERE87], [ROCH88B]. A partir da análise de cada uma, podemos perceber que algumas se encaixam perfeitamente ao ambiente sobre o qual ela está sendo experimentada, mas, para o ambiente focalizado nesta tese, talvez não sejam muito adequadas.

Atualmente podemos observar a existência de um consenso para nossa realidade: o software científico deve ser desenvolvido em duas grandes fases. [ROCH88b] [BAHI88]

A primeira deve ser composta pelo equacionamento do problema, pesquisa da solução, elaboração e teste dos algoritmos necessários à solução numérica encontrada para o problema, e validação desta solução através de um protótipo numérico. Esta validação verificará se os algoritmos podem ser implementados de forma satisfatória, avaliando se a solução do problema é factível.

A segunda consiste da fase de construção do software. É nesta fase que devem ser utilizadas todas as ferramentas da engenharia de software necessárias para o desenvolvimento de um produto final cobrindo as deficiências de documentação, interface com os usuários e manutenibilidade, atingindo um alto grau de qualidade, através da efetivação de todos os atributos de qualidade que foram designados ao software.

A divisão do desenvolvimento do software científico em duas fases é considerado um bom caminho para uma solução, mas não podemos deixar de lado algumas observações.

O fato da primeira fase dispender muito esforço para a validação de um algoritmo numérico complexo, como já foi visto, faz com que os especialistas tendam a considerar o protótipo criado um produto final, considerando desnecessária a realização da segunda fase.

Adicionalmente, se o fator tempo, incluso na segunda fase do desenvolvimento, for muito longo, o algoritmo numérico em questão pode vir a ficar atrás do "estado da arte", ou seja, obsolescer, visto que existe a constante atualização das técnicas usadas nos algoritmos.

Apesar da idéia do software científico ser desenvolvido em duas fases ser compartilhada por algumas pessoas, não existe, ainda, um consenso sobre um ambiente adequado para confecção deste software. Isto porque é difícil definir uma maneira de comunicação entre os especialistas e os analistas e programadores. É difícil definir a distribuição dos profissionais dentro das fases e o mais difícil é promover a efetivação do processo tendo em vista que certamente haverá resistência a estas mudanças, principalmente devido a introdução de normas e conseqüente alteração de uma rotina de anos. Isto tudo ainda é mais dificultado pelo fato do benefício destas mudanças não aparecer imediatamente.

III.5. Ferramentas automatizadas

O aparecimento de ferramentas CASE não alterou sobremaneira este quadro. Principalmente porque ferramentas CASE automatizam técnicas já consolidadas na engenharia de software [SMIT90] e, como já foi mencionado, não existe um padrão sobre métodos e técnicas eficazes para desenvolvimento de software científico. Alguns desenvolvedores fazem uso de ferramentas um pouco mais poderosas (por exemplo, um depurador, um analisador de desempenho, ou ainda, analisadores estáticos e dinâmicos) [WERN92], mas, em geral, os especialistas desconhecem tais ferramentas.

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. [CUNT91]

O ambiente proposto por Cunto é o **FPLUS**, que guia a geração incremental de código, suporta o desenvolvimento com confiabilidade, encoraja um estilo de desenvolvimento uniforme e ajuda o usuário a evoluir gradualmente através de melhores hábitos de desenvolvimento.

TOOLPACK é similar ao **FPLUS** e propõe, também, um ambiente integrado que dê suporte ao desenvolvimento de programas em Fortran. [ILES86]

Rn também é um ambiente projetado para programadores experientes que desenvolvem e mantêm grandes aplicações Fortran. Provê construção de composições de código ("code-composition") avançadas, análise interprocedural sofisticada, compilação incremental e otimizada e monitoramento de execução complexa. [CARL87]

Podemos considerar também como um tipo especial de ferramenta, a biblioteca de rotinas. Existem, atualmente, diversas bibliotecas de software numérico (IMSL, NAG, CERN Library, etc...) e algumas mundialmente utilizadas. Tais bibliotecas são consideradas como um dos melhores exemplos concretos de software reutilizável. [BIGG89] [CHEA89]. Werner juntou a isto o fato do software científico possuir características bem próprias e apresentou uma proposta de estratégia de reutilização para o desenvolvimento deste, com o objetivo de minimizar as dificuldades relacionadas ao seu desenvolvimento.

III.6. Considerações

Através de uma pequena pesquisa realizada com especialistas de um setor de desenvolvimento de software científico da PETROBRAS [WERN88A], Werner identificou que existe uma grande conscientização destes profissionais sobre os problemas de qualidade enfrentados no desenvolvimento de suas aplicações, mas podemos sentir uma pequena resistência no que tange a criação de um grupo de profissionais especializados para transformar seus programas, efetivamente, em softwares produtos.

Os especialistas se mostraram interessados (pelo menos durante a pesquisa) em aprimorar seus conhecimentos em ciência da computação e, por isso, foi desenvolvido um projeto [WERN88B] para elaboração de um guia de desenvolvimento para software

Fortran, detalhando um pouco mais as atividades relacionadas a etapa de pesquisa proposta por Rocha [ROCH88B], especificamente, e ensinando, efetivamente, a utilização de algumas técnicas de engenharia de software (para projeto, codificação em Fortran, teste e controle da qualidade) [GAMA89] para o desenvolvimento de suas aplicações.

Desta forma, podemos concluir que apesar da consciência dos problemas que envolvem o desenvolvimento de software científico e de já existirem algumas propostas elaboradas, o problema fundamental ainda está presente: não existe nenhuma metodologia "padrão" para o desenvolvimento

O fato também da aplicação de uma metodologia para desenvolvimento de software científico não possuir um retorno rápido, em termos de custo, dificulta, sobremaneira, sua implantação e, principalmente, sua experimentação sem a certeza de uma boa adaptação.

É importante ressaltar que a tendência dos algoritmos numéricos, os quais são o núcleo do software científico, é de se tornarem cada vez mais complexos, principalmente com o advento dos computadores que permitem o processamento vetorial e paralelo. Isto irá exigir a elaboração de algoritmos específicos e alteração dos já existentes, aumentando, desta forma, também, as atividades de desenvolvimento de software. Sendo assim, podemos prever, que os problemas que a engenharia de software enfrenta com o desenvolvimento de software científico também tendem a aumentar.

III.7. Conclusão preliminar

Neste capítulo foram expostas as causas e conseqüências do problema que se pretende solucionar, ou pelo menos, amenizar, com o ambiente proposto nesta tese. Foi feita uma descrição do ambiente científico, caracterizando o software desenvolvido nesta área e apontando os problemas enfrentados com a falta de qualidade, em termos de engenharia de software, do produto final gerado.

Vale ressaltar que esta falta de qualidade, por vezes, pode custar milhares de dólares em manutenção nestes ambientes. A tentativa de amenizar este problema é extremamente bem acolhida pelos gerentes destas áreas, apesar de estarem conscientes da dificuldade de implantação de qualquer tipo de mudança.

O próximo capítulo apresenta a técnica de hipertexto que será utilizada no projeto do ambiente proposto.

IV - Hipertexto

Este capítulo apresenta a técnica de hipertexto, descrevendo suas principais características e áreas de aplicação.

IV.1. Histórico

O raciocínio humano parte de um ponto e percorre caminhos não seqüenciais até chegar a alguma conclusão sobre uma questão. Nossa mente se transporta entre fatos do passado, presente e futuro seguindo o processo cognitivo do pensamento. Sistemas hipertexto se aproximam deste processo tendo como principal característica a conectividade e busca das informações de forma não linear.

A idéia de quebrar a seqüência de textos, organizando documentos em trechos e combinando-os conforme a necessidade de organização, já existe, intuitivamente, a muito tempo. Por exemplo, citações, notas de rodapé, referências bibliográficas, quadros e figuras ilustrativos são utilizados intensivamente em documentos.

Vanevar Bush foi o visionário da idéia de hipertexto, apresentando em seu artigo "As we may think" [BUSH45], o projeto do MEMEX, que era um dispositivo para folhear e inserir anotações em uma vasta biblioteca de literatura científica, capaz de conter textos, gráficos, fotografias e desenhos. O MEMEX lançava a idéia de uma rede de documentos conectados por elos, utilizando microfilmes e fotocélulas. [DIPO89]

Bush, desta forma, lança a perspectiva da tecnologia poder ser utilizada para libertar o armazenamento e recuperação das informações, de suas restrições de linearidade.

Um dos primeiros seguidores da idéia lançada por Bush foi Douglas Engelbart, quase vinte anos depois. Engelbart publicou o artigo "A Conceptual Framework for the Augmentation of Man's Intellect" [ENGE63], onde associou a idéia de Bush aos computadores digitais considerando que a evolução do intelecto humano entraria em um novo estágio a partir da utilização de computadores para manipulação automática de símbolos externos.

Em 1968, Engelbart apresentou o sistema AUGMENT (NLS) [ENGE68], colocando em prática suas idéias. Este sistema era uma ferramenta experimental que permitia armazenamento de diversos tipos de documentos: especificações, memorandos, projetos, programas, bibliografias, etc..., propiciando a comunicação e colaboração entre

os pesquisadores no planejamento e desenvolviemento de seus projetos.

Em 1967 foi criado o termo ***HIPERTEXTO***, por Ted Nelson, para descrever a escrita e leitura não linear.

Ted Nelson desenvolveu um projeto para construção de um ambiente literário compartilhado chamado XANADU [NELS68], onde as pessoas do mundo inteiro poderiam manusear idéias e informações generalizadas.

A partir daí, e com o avanço da tecnologia, a idéia de hipertexto começou a influenciar sensivelmente alguns ramos da pesquisa, encontrando-se atualmente em franco processo de desenvolvimento.

Em 1987 aconteceu o primeiro workshop sobre hipertextos. A figura 4.1 apresenta um pequeno histórico de precursores e trabalhos nesta área, até 1987.

1945	Vannevar Bush proposes Memex
1965	Ted Nelson coins the word "Hipertexto"
1967	The Hypertext Editing System and FRESS Brown University, Andy van Dam et al.
1968	Douglas Engelbart demo of NLS system at FJCC
1975	ZOG (now KMS): CMU
1978	Aspen Movie Map, first hypermedia videodisk Andy Lippman, MIT Architecture Machine Group
1984	Filevision from Telos; limited hypermedia database widely available for the Macintosh
1985	Symbolics Document Examiner, Janet Walker
1985	Intermedia, Brown University, Norman Meyrowitz
1986	OWL introduces Guide, first widely available hypertext
1987	Apple introduces Hypercard, Billl Atkinson
1987	Hypertext'87 Workshop, North Carolina

Figura 4.1. Visão da história de Hipertexto [NIEL90]

Após esta data foi cada vez mais crescente o interesse despertado entre os pesquisadores sobre o assunto, tendo sido realizadas até o momento já diversas conferências; vários sistemas comerciais foram desenvolvidos e estão a disposição no mercado; várias propostas foram apresentadas para utilização da tecnologia em diversos ramos e vários trabalhos com aplicações variadas foram publicados. A utilização de diversos tipos de mídias dentro da técnica de hipertexto fez nascer o termo

HIPERMÍDIA despertando ainda mais o interesse nesta linha de pesquisa, que podemos considerar estar hoje em franco processo de desenvolvimento.

IV.2. Definição

Hipertexto utiliza uma forma não sequencial para obtenção de informações textuais, em contraste aos textos tradicionais que são essencialmente linearizados, respeitando uma sequência determinada, ou seja, você primeiro tem que ler a página um, depois a página dois, depois a três e assim por diante.

Hipertexto pode ser definido como sendo um conjunto de trechos de informações textuais conectados entre sí de forma não sequencial.

Na figura 4.2 é apresentado um exemplo. Vamos considerar que iniciaremos a leitura do hipertexto pelo pedaço de texto denominado A. Neste trecho possuímos três opções de lugar pra ir: B, D e E. Se desejarmos ir para E, poderemos então ir para B ou D. Se de E formos para B, podemos escolher ir para C ou F. A escolha dos caminhos é feita pelo leitor do hipertexto.

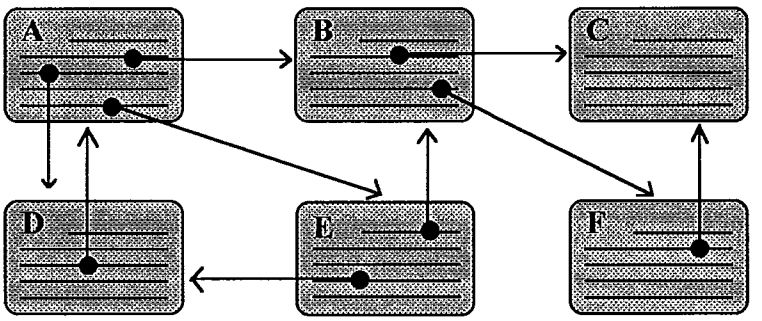


Figura 4.2. Esquema simplificado de um hipertexto com seis nós e nove ligações

Podemos considerar um hipertexto, a nível conceitual, um sistema gerenciador de banco de dados, preservando uma diferença básica: uma base de dados tradicional tem uma estrutura extremamente regular, especificada por uma linguagem de definição de dados, onde todos os dados devem seguir esta estrutura. Uma informação em hipertexto não possui uma definição central e uma estrutura regular. Cada ligação é criada porque faz sentido em termos do conteúdo semântico dos dois nós que ela conecta e não por causa de alguma decisão global.

Isto significa que hipertexto tem uma maior flexibilidade, o que é normalmente uma vantagem, mas pode também ser uma desvantagem [NIEL90], como veremos mais a frente.

Em um nível mais sofisticado podemos considerar hipertexto como um ambiente de comunicação, trabalho cooperativo e aquisição do conhecimento. [LIMA89]

Uma outra aplicação de computador que utiliza um enfoque similar ao de hipertexto são os programas "outliner's". Estes programas são normalmente utilizados para construir sumários de relatórios, ou apresentações, de uma forma hierárquica. Eles conectam informações textuais em um formato definido pelo usuário e, por isso, se aproximam do enfoque de hipertexto. Mas este formato é extremamente restrito a uma hierarquia específica, e esta limitação é o motivo para programas "outliner's" não serem considerados hipertextos. [NIEL90]

IV.3. Conceitos gerais

Hipertexto x Hipermídia

Quando um sistema hipertexto suporta, além de textos, outros meios, ou seja, fotografias, filmes, voz, música, etc..., recebe o nome de hipermídia.

Ser multimídia não é o bastante para um sistema ser hipermídia, mas é perfeitamente possível utilizar bons efeitos dos recursos multimídia como parte de um sistema hipermídia. Hipertexto é uma técnica natural para suportar estas interfaces porque é baseado na interligação entre nós, os quais podem conter qualquer tipo de mídia [NIEL90]. As restrições ou dificuldades existentes serão a nível de hardware.

Vale ressaltar que o termo hipertexto normalmente é utilizado de forma generalizada, incluindo texto e mídia. Neste trabalho consideraremos hipertexto, indistintamente, tanto para sistemas de manipulação apenas de textos, quanto para hipermídia.

Hipertexto x Hiperdocumento

Borges considera sistemas hipertextos ou hipermídias sendo vistos como um grande repositório de informações de diversos tipos: texto, código executável, gráficos, imagens, sons, etc...; fragmentadas segundo algum critério. E chama estes repositórios de hiperdocumentos e o particionamento de nós. [BORG91]

Rada não considera esta diferença, ele mostra que um simples texto se diferencia de um documento que pode conter figuras e gráficos, considerando, desta forma, que o melhor termo para hipertexto, seria hiperdocumento e acrescenta que, na prática, os dois

termos são sinônimos. [RADA89]

Neste trabalho será considerada a conceituação feita por Borges. Hipertexto será considerado o sistema gerenciador do documento (texto), sendo composto por um conjunto de características básicas de implementação. Um hiperdocumento será o conjunto de informações (o próprio documento) que utilizam estes recursos para serem armazenadas e manipuladas.

Usuários de hipertextos

Existem duas grandes classes de usuários de sistemas hipertexto: ***autores*** e ***leitores***. Os autores são responsáveis pela criação do hiperdocumento, a partir da definição dos nós e das ligações entre esses nós.

Os leitores são os usuários que manipulam o hiperdocumento criado pelo autor, através da navegação pelo hipertexto.

Alguns sistemas hipertexto consideram estas duas classes independentes e possuem módulos independentes de autoria e navegação do sistema hipertexto. Outros não consideram esta separação e, por isso, não utilizam este conceito.

IV.4. Arquitetura de sistemas hipertexto

Levando em conta que hipertexto é um tema de pesquisa bem recente, não existe ainda um conjunto padronizado de características que definam a arquitetura física de um sistema hipertexto.

Teoricamente, entretanto, podemos distinguir três componentes básicos que descrevem um modelo de sistema hipertexto considerado como direção para padronização de trabalhos futuros [CAMP88] [CONK87]:

- Uma base de dados de texto;
- Uma rede semântica que conecta os componentes do texto; e
- Uma interface composta por ferramentas para criar e navegar por esta combinação de texto e rede semântica.

IV.4.1. Base de dados de texto

Independentemente das características que diferenciam um sistema hipertexto de outros sistemas computacionais, este possui os mesmos problemas tradicionais de

armazenamento de informações, sendo necessária a existência de um nível de gerenciamento de base de dados para controle de tais problemas.

Neste nível da arquitetura, os nós e ligações do sistema hipertexto são manipulados simplesmente como objetos, não havendo um tratamento especial para estes tipos de informações.

IV.4.2. Rede semântica

É neste nível da arquitetura que o sistema hipertexto estabelece a estrutura dos nós e das ligações que irão manter a relação entre estes nós.

É na definição da rede semântica que se faz efetivamente a diferenciação entre os diversos sistemas de hipertexto. E, por isso, este é o nível mais propício para estabelecimento de uma padronização entre estes sistemas, visando a transferência de informações de um sistema para outro. [NIEL90]

Este nível da arquitetura de um hipertexto é que define a conectividade entre as informações.

A conectividade entre as informações é a característica básica principal e essencial de um sistema hipertexto. É esta a característica que mais diferencia um sistema hipertexto de outros.

Esta conectividade existe segundo diferentes alternativas de ligações entre os nós do hipertexto.

Nó

Um nó em um hipertexto é uma unidade básica de informação. Cada nó pode conter um documento, um trecho de documento ou qualquer conjunto de informação que represente a expressão individual de uma idéia ou conceito. Cabe ao autor do hiperdocumento decidir sobre o conteúdo de um nó.

Não existe um tamanho máximo fixo para um nó, o limite para este tamanho dependerá da implementação do hipertexto. Em alguns sistemas de hipertexto o tamanho do nó pode estar limitado ao tamanho da tela, em outros é possível rolar a tela para apresentação de todo o nó. Em alguns, apenas um nó por vez é apresentado na tela e em outros é utilizado o recurso de janelas múltiplas.

Nós podem ser "tipados" e ter seu conteúdo validado por restrições de integridade ou dirigido por "templates", ou podem não ter tipo associado e aceitar qualquer conteúdo [BIGE88], são os nós "não tipados".

Os sistemas de hipertexto que utilizam nós "tipados" normalmente os diferenciam dentro do sistema através de algum recurso visual (cor, forma, tamanho, etc...).

Os nós também podem ser considerados em conjunto, formando também uma unidade de informação para o sistema. A este agrupamento de nós chamamos nó composto. Um nó composto possui seu próprio nome, tipo e versão.

Ligação

As ligações ou elos de um sistema hipertexto representam os relacionamentos, definidos pelo autor do hiperdocumento, entre os nós. Podem ser consideradas o meio de transporte do hipertexto.

Uma ligação pode estar associada a um nó como um todo ou a parte de um nó.

As ligações existentes em um nó são apresentadas ao usuário no vídeo, normalmente em outra cor ou em vídeo reverso, chamamos este ponto de origem da ligação, que fica marcado no nó, de ***botão*** da ligação. Geralmente o nome deste botão é o nome de seu nó destino.

As ligações também podem ser tipadas, possuindo um conjunto de restrições e atributos que expressem uma relação particular existente entre dois nós. Neste caso, o hipertexto deverá prevêr como apresentar os diferentes tipos de elos ao usuário. Para tanto, pode-se utilizar recursos visuais ou teclas de funções.

As ligações entre nós podem se fazer em apenas um sentido ou em ambos os sentidos, dependerá da implementação do sistema.

Um sistema hipertexto pode suportar ligações entre os nós formando uma estrutura hierárquica ou formando uma estrutura não hierárquica.

A estrutura hierárquica é formada quando a ligação entre os nós é feita de forma organizacional e a principal vantagem desta estrutura é a simplicidade dos comandos para percorrer o hipertexto, o que também ajuda a diminuir o problema de desorientação, ou seja, do usuário perder a noção de localização e direção dentro do hipertexto (figura 4.3). A principal desvantagem desta organização é a dependência da

sua estrutura ao padrão específico que foi utilizado para criá-la. [LIMA89]

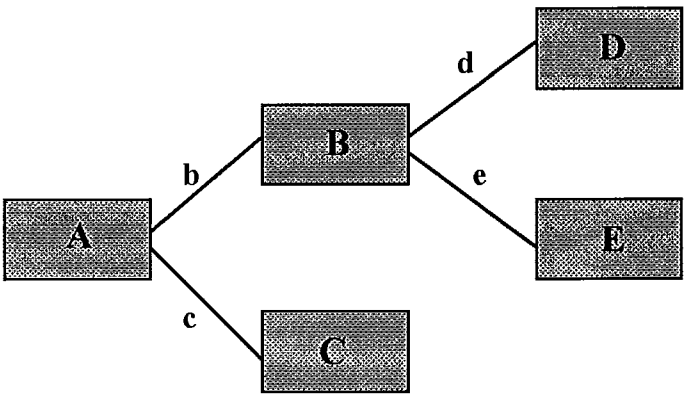


Figura 4.3. Exemplo de ligação hierárquica

Uma estrutura não hierárquica é formada quando são utilizadas ligações referenciais para conectar os nós. Uma ligação referencial possui um ponto em um nó, como origem, e um ponto ou região de um outro nó, como destino (figura 4.4).

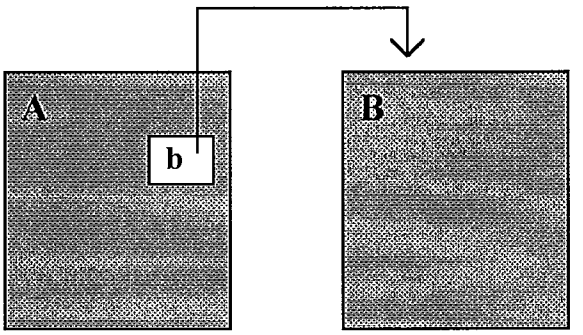


Figura 4.4. Exemplo de ligação referencial

IV.4.3. Interface com o usuário

O nível interface da arquitetura de um sistema hipertexto, se preocupa com questões relevantes sobre como apresentar ao usuário as informações definidas no nível anterior, onde foi criada a estrutura semântica do sistema.

Neste nível são utilizadas ferramentas para definição e implementação de recursos para solucionar questões do tipo: quais comandos deverão estar disponíveis para o usuário, como serão apresentados os nós e as ligações, que tipos de diagramas deverão ser apresentados ao usuário, etc....

IV.5. Principais características

Além dos componentes que compõem a arquitetura de um sistema hipertexto, existe um conjunto de características básicas, algumas essenciais, outras desejáveis, para um sistema ser considerado um hipertexto ou, pelo menos, para ser considerado um bom sistema hipertexto. Estas características básicas podem ser definidas como sendo as seguintes: [LIMA89]

- Interface amigável;
- Acesso compartilhado;
- Visualização do conjunto de informações;
- Recuperação e busca de dados;
- Controle de versões.

IV.5.1. Interface amigável

A qualidade de um sistema hipertexto depende sobremaneira da funcionalidade das ferramentas oferecidas aos usuários, tanto para criação quanto para navegação do hiperdocumento.

Uma interface simples e bastante amigável é essencial para o sucesso de um hipertexto.

De preferência, deve ser utilizado o mesmo conjunto de ferramentas integradas, tanto para autores quanto para leitores do hipertexto, caso o sistema utilize estes conceitos, propiciando o trabalho cooperativo de criação e uso do sistema.

IV.5.2. Acesso compartilhado

Permitir a criação de um ambiente de trabalho cooperativo é plenamente possível e viável em sistemas hipertexto através da implementação de acesso multiusuário.

Este tipo de característica é bastante desejada em um hipertexto, para propiciar atividades de discussão, decisão, acordo, troca de idéias e outras, dentro de um hiperdocumento. Mas, vale ressaltar, que para garantir a consistência das informações manipuladas, outros mecanismos devem ser implementados juntamente, assim como, controle de concorrência, controle de versões, visões, etc....

IV.5.3. Visualização do conjunto de informações

Devido ao risco existente dos usuários (leitores) de um hipertexto ficarem desorientados, ou terem problemas em encontrar a informação que eles precisam, é fundamental a existência de ferramentas para visualização da estrutura da rede de informações e de contextos individuais contidos nesta estrutura.

Estas ferramentas devem gerar diferentes tipos de mapas e diagramas, ilustrando as conexões e contextos existentes no hipertexto. [LIMA89]

O mapa global deve apresentar todos os nós e ligações existentes no hipertexto (figura 4.5). O mapa regional deve ilustrar os nós e ligações de uma determinada região do hipertexto. E o mapa parcial deve apresentar as ligações de um nó específico (figura 4.6). Alguns sistemas permitem a navegação pelo hipertexto através de mapas, obtendo com isso, acesso mais rápido ao nó desejado. [BORG91]

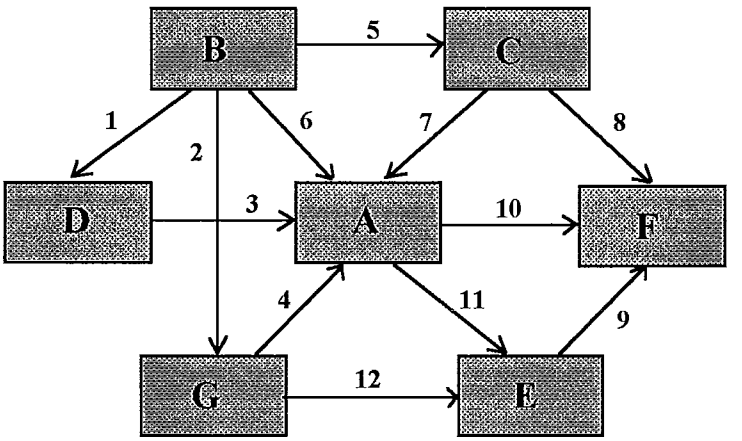


Figura 4.5. Exemplo do mapa global de um hiperdocumento

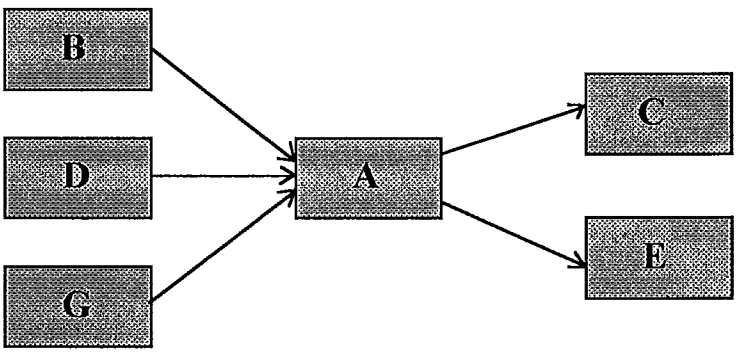


Figura 4.6. Exemplo do mapa parcial de um nó denominado A

Outro recurso utilizado é a trilha. Uma trilha contém a sequência de nós percorridos, permitindo ao leitor uma melhor orientação no hiperdocumento.

Excursões são trilhas guardadas e são utilizadas como orientação na busca de um conhecimento que já foi explorado anteriormente.

IV.5.4. Recuperação e busca de dados

Um sistema hipertexto de qualidade precisa possuir outras formas de busca e recuperação de dados além do método de navegação.

Uma dessas outras formas, como já foi mencionado no item anterior, é a navegação pelo hipertexto através de mapas. Esta facilidade torna a recuperação de informações muito mais rápida e eficaz.

Assim como em um sistema gerenciador de banco de dados tradicional, também podemos efetuar buscas em um hipertexto através de palavras-chaves, mas para isso é preciso associar a cada nó uma ou mais palavras-chave. Ou procurar uma determinada informação (por exemplo, uma cadeia de caracteres) nos nós do hiperdocumento.

Outra forma de recuperação de dados é o mecanismo de visões ou filtros. Através deste mecanismo é possível que o autor defina uma máscara, a qual irá limitar e/ou definir os nós que poderão ser acessados pelo leitor, permitindo a ocultação de informações que estejam fora do escopo da utilização desejada.

Este mecanismo também permite garantir a segurança das informações, restringindo o acesso ao hipertexto.

IV.5.5. Controle de versões

Esta característica é de grande utilidade para sistemas hipertexto porque permite preservar diversas versões históricas da criação do hiperdocumento. É uma característica essencial em sistemas hipertextos que sejam usados no suporte ao ambiente de trabalho cooperativo.

O objetivo do histórico de versões é documentar as diversas etapas envolvidas na concepção de um objeto de projeto, organizando versões através de um relacionamento de derivação, sucessão e eventualmente equivalência. [PRIC89]

IV.6 Algumas áreas de aplicação de hipertexto

Hipertexto não precisa, necessariamente, fazer parte de um sistema hipertexto como tal. É plenamente possível utilizar idéias de hipertexto e integrá-las em outros

sistemas computacionais. [NIEL90]

Este item irá apresentar algumas áreas de aplicação de hipertexto considerando tanto a possibilidade de criação de sistemas hipertexto, quanto a viabilidade de utilização da técnica de hipertexto dentro de outros sistemas computacionais, nestas áreas.

Vale ressaltar que não são todas as aplicações computacionais que podem ser desenvolvidas em hipertexto. Shneiderman propôs o que chamou "as três regras de ouro", para determinar se uma aplicação é condizente com o uso de hipertexto ou não. Estas regras são as seguintes: [SHNE89]

- um grande corpo de informação é organizado em numerosos fragmentos;
- os fragmentos se relacionam entre sí;
- o usuário precisa de somente pequenas frações, de cada vez.

IV.6.1. Aplicações na Engenharia de Software

A engenharia de software se preocupa basicamente em dar suporte ao Processo de Desenvolvimento de Software propondo métodos, técnicas e ferramentas para geração de software de qualidade. É sabido que a engenharia de software, entretanto, enfrenta diversos problemas para atingir seus objetivos, apesar do advento das ferramentas CASE: [BORG91]

- o empirismo dos métodos não consegue eliminar as ambiguidades até a versão computacional, e estas acabam sendo resolvidas pelo próprio programador, o qual tem menor visão global do sistema;

- a falta de formalização no processo de transformação entre as diversas fases do desenvolvimento faz com que tal processo esteja muito propício a erros;

- a falta de correlação entre os produtos resultantes destas transformações faz com que a única versão utilizável do software seja o conjunto de códigos fonte. Por causa disto, a etapa de manutenção, quase sempre, se preocupa em atualizar apenas os programas fontes, aumentando muito, a possibilidade de erros;

- a dificuldade de representação das intenções do usuário e mesmo a indefinição por parte este fazem com que a versão final deixe de atender às suas reais intenções;

- a falta de ferramentas eficazes para coordenação do trabalho de equipe faz com que existam graves divergências entre as visões dos diversos integrantes do projeto.

Alguns métodos propostos pela engenharia de software tentam resolver tais problemas, como por exemplo, Métodos Formais, Transformações Automáticas e Métodos Evolutivos [KEMM90], mas nenhum, realmente, conseguiu, até hoje, solucioná-los.

O processo de desenvolvimento de software segue etapas padrões independentemente do paradigma de engenharia de software utilizado. Estas etapas são ultrapassadas, uma a uma, como uma sequência de transformações até chegar a uma versão executável do software em um meio computacional. Os produtos gerados em cada uma dessas etapas são documentos altamente relacionados entre sí (especificações, códigos, relatórios de testes, etc...).

Outras atividades também são bastante intensas no processo de desenvolvimento de software, como troca de idéias entre os integrantes do projeto, controle das atividades de cada um desses integrantes, controle de consistência entre os documentos, etc....

Se relacionarmos as três regras de Shneiderman [SHNE89] a estas atividades, constatamos que estas são plenamente compatíveis com a idéia de hipertexto: possuem um grande corpo de informações, organizadas em numerosos fragmentos que se relacionam entre sí e são utilizados por parte.

Alguns autores vão mais longe e chegam a afirmar que "a utilização do enfoque de hipertextos será de fundamental importância para que a engenharia de software alcance seus objetivos." [BORG92]. A principal atividade do processo de desenvolvimento de software a se beneficiar com a utilização de hipertextos é a atividade de documentação do software. A grande vantagem que a utilização deste enfoque pode trazer, é a possibilidade de relacionar os diversos elementos que compõem cada documento e navegar entre eles. Desta forma, é possível ligar, por exemplo, um DFD da Especificação de Requisitos com um DEM do Projeto da Arquitetura, que por sua vez poderia estar ligado a uma rotina em código fonte.

O hipertexto também possibilitaria o controle de versões de cada uma dessas partes, bem como a restrição de acessos. A principal vantagem alcançada é a grande melhoria das condições de manutenção do software, possibilitando uma sensível redução dos custos nesta etapa.

Além da grande utilidade de hipertextos em todo o processo de desenvolvimento de software, também podemos apontar grandes vantagens na utilização deste enfoque em cada uma das etapas, separadamente.

Por exemplo, se considerarmos apenas a etapa de codificação, programas são extremamente ricos em conectividade. Uma variável é definida no cabeçalho do programa e é utilizada em diversos pontos do código. Se este programa estiver sob a gerência de um hipertexto, é possível associar cada aparição da variável a sua definição, desta forma, quando o leitor desejar este tipo de informação, basta pressionar uma tecla, para que uma janela com a definição apareça na tela [RADA89]. O mesmo pode acontecer com chamadas a rotinas, comentários, conteúdo de arquivos, etc....

A atividade de coordenação de trabalho em equipe também pode se beneficiar muito da utilização deste enfoque. Um trabalho em equipe tem como principal característica, a troca de informações. Esta troca pode ser formal, transmitida na forma de documentos, ou informal, quando são feitas exposições em reuniões ou conversas. É extremamente difícil formalizar ou organizar toda informação informal gerada. Entretanto, a filosofia de hipertextos de permitir a livre associação de informações e idéias, nos leva a acreditar que essa combinação de informação formal e informal pode ser plenamente gerenciada por um sistema hipertexto, onde as informações poderão ser geradas e compartilhadas por cada um dos integrantes da equipe, nas várias etapas do processo de desenvolvimento de software.

A primeira etapa do processo de desenvolvimento de software é marcada pela atividade de levantamento de dados. Esta atividade consiste em recolher informações de diversos tipos, entrevistas, documentos, mapas, relatórios, formulários, etc..., para gerar um documento formal contendo os resultados desta etapa. A geração deste documento não é muito fácil, principalmente se houver um grupo de pessoas realizando este trabalho.

A utilização de hipertextos, ou melhor, hipermídia, para tratamento destes dados também é bastante viável, podendo tornar esta atividade mais produtiva e eficaz. Primeiro porque permite o armazenamento de diferentes meios. Segundo porque permite associar livremente estes dados de modo a interligar o conjunto de informações possibilitando o entendimento global do que foi levantado. E, terceiro, porque representará o ponto de partida do processo de documentação, que irá se associar no futuro às outras versões geradas pelo processo de desenvolvimento [BORG91].

Existem diversos projetos sendo realizados com o objetivo de dar suporte a todo, ou a parte, do processo de desenvolvimento de software, utilizando hipertexto. Os

principais, encontrados na literatura, são os seguintes:

Dynamic Design: [BIGE87] Este trabalho está baseado em um projeto desenvolvido pelo Grupo de Automação de Projetos da Tektronix Inc. Este grupo desenvolveu o DynamicDesign.

DynamicDesign é um ambiente CASE para a linguagem C baseado em Hipertexto. Ele armazena toda a documentação do programa (requisitos, documentação de projeto, manual do usuário, notas de implementação, etc...) juntamente com o grafo do código fonte que é criado através de um utilitário chamado GraphBuild. Este tem por função converter o código fonte C para um grafo fonte em hipertexto a partir da árvore de chamada do programa. Este utilitário também cria um dicionário de dados para o programa contendo todas as variáveis locais e globais declaradas no programa.

DynamicDesign tem como base para o seu desenvolvimento o modelo Hypertext Abstract Machine (HAM) [CAMP88] desenvolvido também pela Tektronix, Inc. HAM é um modelo de dados para armazenamento do hipertexto baseado em transação, que permite estruturação arbitrária da informação e controle de versões, tanto da informação quanto da estrutura. DynamicDesign e HAM são considerados independentes.

DIF: Este projeto foi desenvolvido pela Universidade da Southern California. É um sistema hipertexto que ajuda a integrar e gerenciar os documentos produzidos e usados através do ciclo de vida de projetos de software [GARG89].

Este projeto faz parte de um projeto maior chamado "System Factory", que vem a ser um laboratório experimental criado para estudo do desenvolvimento, uso e manutenção de software de grande escala. [SCAC89]

DIF permite a integração de ferramentas de apoio ao processo sob o ambiente UNIX; permite a reutilização de partes de diferentes documentos, dá suporte a vários projetos sob o mesmo hipertexto, efetua controle de versões dos documentos, etc....

Design Journal: [BEGE88] [CONK87] Este é um projeto experimental da MCC Software Technology Program e visa produzir um ambiente de desenvolvimento de software capaz de mediar as decisões de projeto, tanto a nível individual como em grupo.

gIBIS (graphical IBIS) é um protótipo do projeto Design Journal e automatiza o método IBIS (Issue-Based Information Systems). Este método consiste em reunir projetistas, usuários e implementadores para resolver questões de projeto. Cada questão implica em uma ou mais posições, que por sua vez, podem ser suportadas ou

contrariadas, por um ou mais argumentos. [BEGE88]

Este sistema é utilizado para tentar resolver problemas em se lidar com grandes volumes de informações não estruturadas e o processo de tomada de decisão dentro de um projeto de software.

Rigi: Este projeto tem como objetivo dar suporte aos documentos de projeto, em todas as etapas do ciclo de vida, sevindo como meio de comunicação e coordenação entre os integrantes do projeto.

Os três prinicipais problemas que o Rigi tenta atacar são os seguintes: o domínio da comlexidade estrutural de grandes sistemas; a apresentação efetiva de toda a informação acumulada durante o desenvolvimento; e a definição de procedimentos para verificar e manter a completeza, a consistência, e a rastreabilidade das descrições do sistema. [MULL88]

IV.6.2. Aplicações em outras áreas

IV.6.2.1. Associação a outras áreas da computação

Inteligência artificial

O desenvolvimento de um sistema especialista envolve duas atividades principais: A aquisição do conhecimento e a criação da base de conhecimento.

O trabalho de decidir sobre quais conceitos colocar em cada nó e como relacionar os nós ao compôr um hiperdocumento é semelhante ao trabalho de aquisição do conhecimento sobre um certo domínio [THOM87].

Também na representação do conhecimento, podemos utilizar hipertexto devido a possibilidade de implementar tanto redes semânticas, quanto frames e regras dentro da arquitetura de um hipertexto [CONK87] [THOM87].

Banco de dados

A associação com a tecnologia de banco de dados criou o que chamamos de HIPERBASE [SCHU91]. Existem diferentes formas de enfoque para implementação de uma hiperbase. A primeira é considerar o hipertexto apoiando o SGBD, a segunda é considerar o SGBD dando suporte ao hipertexto e a terceira é criar um sistema unindo as duas tecnologias sem que prevalesça as características de implementação de uma.

IV.6.2.2. Aplicações em outras áreas da computação

Documentação "on-line"

Diversos pacotes de software mais recentes estão utilizando manuais on-line ou sistemas de help on-line na forma de hipertexto. Este tipo de documentação é considerada a mais natural de todas as aplicações em hipertexto [NIEL90].

Outras aplicações de documentação de apoio ao usuário podem estender esta facilidade de hipertexto, por exemplo, tutores e mensagens de erro.

Interface com o usuário

A abordagem dada por alguns sistemas de gerência de interface com o usuário é bem similar a defendida nas propostas de hipertexto dinâmico, no qual ações são associadas aos nós ou aos elos da rede. Ambos compartilham a ênfase na facilidade de interação entre os sistemas e as pessoas. [DIPO89]

IV.6.2.3. Aplicações na educação

Ensino

A área de ensino talvez seja, realmente, uma das áreas que mais possa se beneficiar das características que o enfoque de hipertexto possui, levando em conta os diversos sistemas de hipertextos para ensino que têm sido desenvolvidos, mostrando uma grande e crescente aceitação por parte tanto dos alunos, quanto dos professores.

A possibilidade de conectividade de informações de diferentes tipos - texto, vídeo, desenhos, música ou outra informação - juntamente com o grau de liberdade permitido para percorrê-las, justificam o grande sucesso do enfoque de hipertexto (hipermídia) para desenvolvimento de sistemas de apoio educacional.

Se considerarmos um sistema de hipertexto que permita ao aluno o acesso a uma grande quantidade de informações, visualizando todo conjunto, e a partir destas relacioná-las ao assunto que está sendo estudado, este aluno poderá adquirir um considerável senso de organização de idéias, bem como, um completo entendimento sobre as causas e/ou fatores que propiciam a ocorrência de um determinado fenômeno, podendo estar apto para entender fenômenos cada vez mais complexos.

Bibliotecas

Podemos considerar uma biblioteca como o principal setor que envolve arquivamento e manipulação de documentos de diversos tipos (livros, artigos, mapas, desenhos, etc...)

Podemos considerar, inclusive, que, em termos funcionais, uma biblioteca utiliza o enfoque de hipertexto: uma pessoa chega, consulta o fichário que possui relação de todas as obras, com títulos, autores e pequeno resumo, encontra a referência da obra que deseja, vai na estante e encontra. Durante a pesquisa nesta obra, pode ser encontrada necessidade de consulta a uma outra obra. Então a pessoa vai novamente ao fichário verifica a refência desta outra obra, vai até a estante consulta, pode nesta segunda consulta precisar buscar uma terceira, e depois retorna a obra anterior.

O benefício que a automação de uma biblioteca utilizando um sistema de hipertexto (hipermídia) traz, é bastante evidente e intuitivo, inclusive, o nascimento da idéia de hipertexto se deu através do Memex, de Vanevar Bush, como já foi citado no item I. E, a partir daí, muitos outros sistemas de hipertexto (hipermídia) já foram desenvolvidos com objetivo de recuperação/atualização de informações em bibliotecas, todos eles obtendo bastante sucesso.

IV.6.2.4. Aplicações na medicina

O hipertexto pode ser utilizado na área médica, principalmente, para facilitar o acesso às informações sobre um paciente, de uma forma compartilhada, rápida e eficiente. [FRIS88]

Esta facilidade seria importante porque um médico, muitas vezes, precisa ter acesso aos registros de pacientes anteriormente tratados de doenças semelhantes, publicações recentemente feitas sobre o assunto, manuais e livros de medicina, além de radiografias e exames do paciente. Para tanto, o médico dispenderia muito tempo e, é neste ponto, que o hipertexto entra como uma técnica para agilizar a recuperação destas informações.

IV.7. Exemplos de hipertexto

Podemos considerar os sistemas hipertexto existentes divididos em duas gerações. [HALA88]

A principal diferença entre os sistemas da primeira geração para os da segunda,

está nos recursos utilizados para interface com o usuário. O avanço tecnológico e o surgimento de sistemas baseados em estações de trabalho delimitou, mais ou menos em 1980, estas duas gerações.

A geração atual utiliza recursos gráficos e outros meios de representação de diferentes tipos de informações, como som, animação, etc..., além de oferecerem mecanismos mais eficientes de apoio ao usuário, como diagramas gráficos para visualização e percurso da rede de nós e ligações das informações do sistema [LIMA89].

Podemos considerar como sistemas pertencentes a primeira geração o Xanadu, o NLS/Augment, o FRESS e o ZOG. Os dois primeiros já foram citados no item I deste trabalho.

A geração atual possui um número muito grande de sistemas prontos e em fase de pesquisa. Podemos citar como exemplos de sistemas hipertexto da geração atual o NoteCards [HALA88], Intermedia [CAMP88], HyperCard [GOOD87], Guide [BROW87], KMS [AKSC88], Hyperties [SHNE89]. O Xanadu vem sofrendo diversas alterações para estar sempre atualizado em termos tecnológicos, por isso a versão atual deste também é considerada pertencente a segunda geração de sistemas hipertexto.

NoteCards: Foi projetado para auxiliar as pessoas no processamento de suas idéias e no desenvolvimento de melhores modelos conceituais para a análise e solução de problemas.

O NoteCards foi implementado em uma máquina da xerox que faz parte de um ambiente de programação Lisp e que possui poderosas estações de trabalho com vídeos de alta resolução gráfica. [CONK87]

Intermedia: Tem como objetivo auxiliar professores e alunos em suas atividades no ambiente educacional e de pesquisa. Os professores podem organizar e apresentar suas lições e os alunos podem estudar a matéria criando suas próprias anotações e comentários.

O Intermedia foi implementado sobre o sistema operacional 4.2 BSD UNIX e é executado no IBM RT/PC e na estação SUN.

HyperCard: É um sistema hipermídia projetado para armazenar e recuperar informações de diferentes tipos, como palavras, gráficos, desenhos e fotografias digitalizadas, sobre os mais variados assuntos. É para ser utilizado como um ambiente de informações no Macintosh da linha Apple.

Guide: Este sistema oferece ferramentas simples e interativas para utilização de hipertextos em ambientes mais restritos. Guide foi o primeiro sistema de hipertexto comercial popular. Sua primeira versão foi para o Macintosh, mais tarde foi também implementado para o IBM PC. A interface com o usuário é exatamente a mesma nos dois computadores.

KMS: O KMS é o sucessor do ZOG e é, provavelmente, um dos maiores sistemas hipertexto existente. O KMS é um sistema hipermídia distribuído em diversas estações de trabalho, que provê assistência as atividades de um grupo de pessoas trabalhando cooperativamente. O ZOG rodava em computadores de grande porte, atualmente, o KMS é utilizado em workstations PERQ.

Hyperties: Foi projetado para permitir a exploração de recursos de informação de uma forma fácil e agradável. Hyperties está disponível como um produto comercial para microcomputadores IBM PC.

Foram apresentados alguns sistemas hipertexto mais conhecidos e amplamente divulgados. Todos, com exceção do Intermedia que possui um propósito específico, são considerados sistemas para uso geral.

Diversos outros sistemas, dando suporte a outras áreas, existem disponíveis no mercado ou em fase de pesquisa. Alguns exemplos são os seguintes: o Textnet [TRIG83] como sendo um sistema de apoio a bibliotecas on-line; o FileVision e Planetext também para uso geral.

IV.8. Considerações sobre hipertexto

As vantagens adquiridas com a utilização do enfoque de hipertextos são evidenciadas pelas desvantagens de utilização dos enfoques tradicionais e podem ser resumidas abaixo: [CONK87]

- facilidades para seguir referências;
- facilidades para criar novas referências;
- estruturação da informação em hierarquias simples, múltiplas, ou em redes;
- visões globais através de mapas gráficos, sumários e índices, facilitando a reestruturação de documentos complexos;

- documentos personalizados encadeando trechos de texto em várias formas possíveis, em função do interesse dos distintos usuários;

- modularidade da informação levando a reutilização de trechos em múltiplos nós e documentos, com menor redundância e inconsitência;

- consistência da informação referenciada, pois os elos relativos a um trecho acompanham-no caso seja movido dentro do nó/documento, ou para fora deles;

- empilhamento de tarefas, permitindo ao usuário trabalhar, concorrentemente, em um ou mais nós do documento, podendo a qualquer tempo retomar as atividades passadas;

- colaboração entre vários autores (e leitores) na elaboração e revisão de documentos.

Até o momento não citamos desvantagens na utilização deste enfoque, entretanto, alguns problemas sérios existem e precisam ser bem entendidos pelos usuários de hipertextos.

Quando um hiperdocumento é gerado, os nós que deverão compôr o sistema são criados e as ligações entre estes nós são definidas pelo autor do hiperdocumento. O usuário do hipertexto necessariamente precisa percorrer o sistema através da estrutura de rede que representa o hiperdocumento, composta pelos nós e suas ligações, utilizando os "links" criados pelo autor como passagem de um nó para o outro.

O fato deste estar preso unicamente a esta representação em rede de nós e com os "links" pré-estabelecidos pelo autor do sistema, citado por Frank Halask como a "Tirania das Ligações" [HALA91], é um dos maiores problemas enfrentados pelos sistemas de hipertexto.

O tempo gasto na autoria de um hiperdocumento também é considerado um ponto de desvantagem na utilização deste enfoque.

A liberdade de busca das informações também pode causar problemas de desorientação do usuário durante a utilização do hipertexto, promovendo o descontrole deste sobre o trabalho que estiver sendo realizado. Por isso, algumas características de sistemas hipertexto (mapas, excursões, trilhas, etc...), descritas neste capítulo, são consideradas essenciais para amenizar este problema.

Outros problemas encontrados na utilização deste enfoque têm origem nas mesmas características responsáveis pelo poder do hipertexto: a liberdade de registrar e associar idéias, durante a autoria; e a liberdade de escolher por onde começar e quais referências seguir, durante a leitura. [DIPO89]

Estes sistemas, em alguns casos, podem ser muito rígidos apesar desta palavra parecer ir contra toda a propaganda feita sobre hipertextos.

IV.9. Conclusão preliminar

Este capítulo apresentou um pequeno histórico sobre a técnica de hipertexto, alguns conceitos gerais e as características básicas que compõem um sistema hipertexto. Mostrou algumas áreas de aplicação e alguns exemplos de sistemas hipertexto desenvolvidos. Detalhou um pouco mais a aplicação de hipertexto na engenharia de software e apresentou, dentre outros sistemas, o Dynamic Design que foi o projeto tomado como ponto de partida para o desenvolvimento deste trabalho.

O próximo capítulo descreverá o ambiente proposto neste trabalho que utiliza um sistema hipertexto como principal ferramenta.

V - Ambiente proposto

Este capítulo apresenta o ambiente proposto nesta tese, descrevendo os objetivos e uma especificação informal do ambiente.

V.1. Introdução

Em ambientes científicos, como já foi visto no capítulo 3, o software normalmente é desenvolvido pelo próprio especialista da área. Este especialista, na maioria dos casos, não possui formação em engenharia de software, apenas utiliza-se da programação (codificação) como uma ferramenta para sua atividade fim.

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.

Acreditamos que 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.

Isto porque, primeiro, o software desenvolvido pelo especialista da área científica, de uma maneira geral, é extremamente dependente de conhecimento específico da área, tornando muito trabalhoso, custoso e não confiável, a transmissão de tal conhecimento do especialista para o analista ou programador. Segundo, o processo de conscientização dos profissionais da área é muito demorado devido a rejeição a mudanças inerente a todo ser humano, principalmente, se tratando de uma mudança tão radical. Terceiro, a inclusão de profissionais de engenharia de software no processo de desenvolvimento aumentará extremamente o custo do software que, apesar de justificado através de uma boa relação custo/benefício, normalmente reprime a empresa em sua implantação imediata, levando em conta que o retorno de tal investimento é a médio e longo prazo.

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

Esta tese 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 principal vantagem, para o especialista, na utilização deste ambiente é que este será induzido a utilizar o ambiente a partir de diversas facilidades que lhe serão oferecidas, diminuindo a fator resistência, normal em qualquer situação de 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.

Como já foi visto no capítulo 2 deste documento, para ser realizada qualquer modificação em um programa, o programador gasta um tempo razoavelmente grande para, simplesmente, analisar e entender o código. Como foi citado, uma pesquisa recente mostrou que o estudo e análise do código consome em média 47 % do tempo total gasto na manutenção realizada.

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, ou mais, para entender seu próprio programa.

Sendo assim, pretendemos, através da utilização do enfoque de hipertextos no HyFor, em ambientes científicos, 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.

A afirmativa acima se apoia na explanação feita no item anterior, sobre as principais vantagens e importância da utilização de hipertextos na engenharia software.

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

Como já foi visto, tão ruim quanto não haver documentação, é existir uma documentação não atualizada sobre um programa. Desta forma, consideramos que,

sendo precária ou inexistente a documentação de programas em ambientes científicos, a geração automática de uma documentação que seja atualizada, consistente e completa ao que se propõe, mesmo sendo simplificada, conseguirá aumentar a qualidade final de um novo programa gerado e também auxiliará sobremaneira o entendimento do programa na fase de manutenção, diminuindo, conseqüentemente, o custo total desta fase.

O ambiente também permitirá a criação de versões e um controle de alterações realizadas no código, possibilitando a geração de um relatório de manutenção sobre o programa que permitirá um melhor acompanhamento desta fase.

Além das funções citadas acima, foram criadas outras funções no ambiente que também auxiliarão o desenvolvimento de programas com um maior nível de qualidade. As outras funções disponíveis são: geração de um relatório de avaliação da complexidade do programa; geração de um relatório de avaliação da qualidade do código; e reestruturação do código.

Mais a frente todas as funções citadas acima serão explicadas com mais detalhes.

V.2. Especificação informal do HyFor

O ambiente proposto promove a integração de ferramentas que juntas possibilitarão alcançar os objetivos descritos acima.

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.

Abaixo serão descritas com mais detalhes cada uma dessas ferramentas.

V.2.1. ACF

O ACF possui quatro módulos:

- Avaliador da complexidade do código;
- Avaliador da qualidade do código;
- Reestruturador 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.

V.2.1.1. Avaliador da Complexidade

Como já foi visto no capítulo 2, o modelo de Rocha para avaliação da qualidade de software pode ser aplicado nas seguintes fases do seu desenvolvimento: especificação, projeto e codificação de programas.

Neste trabalho iremos considerar medidas (ou métricas) associadas à avaliação da qualidade na fase de codificação, ou seja, serão avaliados os programas desenvolvidos.

O apêndice A apresenta uma relação das métricas de complexidade encontradas na literatura e passíveis de serem selecionadas para análise e as que foram consideradas neste trabalho. Uma descrição mais detalhada pode ser encontrada em [BAHI88].

Este módulo do ambiente se baseia quase que integralmente no trabalho desenvolvido por Bahia [BAHI92]. Neste foi considerado que as métricas de complexidade são as que devem ser abordadas pois permitem, de maneira geral, que seu processo de medição seja automatizado.

Neste módulo iremos realizar 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.

Vale ressaltar que serão apresentados apenas valores quantitativos das métricas,

este módulo não prevê a obtenção de dados de erros e custos e a automatização de correlações e regressões estatísticas. Este trabalho adicional foi desenvolvido por Bahia [BAHI92] alcançando bastante sucesso.

O relatório de avaliação da complexidade do código, resultante da análise do ACF poderá, se desejado, ser considerado um nó do hiperdocumento. Este nó estará ligado ao nó do módulo principal.

O apêndice B apresenta um exemplo deste relatório.

V.2.1.2. 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" [PETR90]. 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 a manutenibilidade e 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.

O ACF avalia os seguintes pontos:

- **Modularização**

Um programa muito longo fica pouco legível, principalmente quando há um grande número de IF's e laços aninhados. O avaliador de qualidade verifica o tamanho de cada módulo que compõe o programa, utilizando como tamanho máximo para um módulo 100 linhas.

Vale ressaltar que não existe um consenso sobre o tamanho ideal para um módulo, esta análise é feita sobre um tamanho máximo considerado, no ambiente científico, um tamanho razoável, e caso o módulo ultrapasse este tamanho o avaliador

emite uma mensagem de alerta avisando que o módulo pode não estar constituído da implementação de uma função única e bem definida.

- **Documentação**

O avaliador de qualidade considera o cabeçalho, contendo informações gerais sobre um módulo, fundamental para a documentação de um programa. Ele avalia para cada módulo a existência de um cabeçalho dentro dos moldes definidos pelo ACF. O conteúdo das informações não é verificado.

- **Indentação**

A utilização de indentação quando existem comandos aninhados no código, aumenta extremamente a legibilidade do programa. O avaliador de qualidade verifica a utilização de indentação para os comandos IF e DO.

- **Linhas de comentário**

O avaliador verifica se existem linhas de comentários entre linhas de continuação de comandos, considerando que além de ser confusa a visualização de um comando interrompido por comentários, algum software utilitário pode alterar a disposição dos comentários deixando-os sem sentido.

Também é verificado se existem linhas de comentários entre o comando END e a declaração de um subprograma, pois na listagem de compilação, podem ficar separadas do subprograma a que elas se referem. As linhas de comentário devem estar dentro da unidade de programa a que pertencem.

- **Nomes mnemônicos**

O avaliador de qualidade considera que nomes de operandos Fortran devem ser curtos visando facilitar a depuração e a manutenção, e nomes com mais de seis caracteres prejudicam a portabilidade do programa.

- **Palavras reservadas**

Como o Fortran não possui palavras reservadas, é possível utilizar nomes de comandos ou palavras chaves como nomes de operandos em um programa. Isto dificulta sobremaneira a legibilidade do programa e deve ser completamente evitado.

- **Ordenação dos labels dos comandos**

Ë considerado que labels usados ao longo do programa devem estar ordenados de forma crescente para facilitar a leitura do programa.

- **Declaração de variáveis**

O avaliador verifica se todas as variáveis utilizadas ao longo do programa foram declaradas.

- **Comando DIMENSION**

Vetores devem ser dimensionados através de uma declaração de tipo ao invés de se utilizar o comando DIMENSION.

- **Definição de blocos COMMON**

O avaliador de qualidade verifica se a declaração de um bloco COMMON foi feita da mesma forma em todos os módulos do programa que utilizam o COMMON.

Qualquer descuido na utilização deste comando pode provocar problemas seríssimos de propagação de erros durante a manutenção do programa.

- **Laços de DO**

O avaliador de qualidade verifica se cada laço de DO termina com um comando CONTINUE próprio. Laços de DO aninhados devem ter tantos CONTINUE quantos forem os laços de DO. Isto facilita a visualização de cada laço de DO e permite a inserção de comandos ao final de um determinado laço.

O apêndice C apresenta um exemplo deste relatório.

V.2.1.3. Reestruturador

O objetivo deste módulo é aumentar a legibilidade e modulariade do programa, através da utilização de uma técnica de reestruturação de código.

Este módulo não será especificado com mais detalhes neste trabalho.

V.2.1.4. 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 estática 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 melhor flexibilidade e alterabilidade, para a utilização de diversos sistemas hipertexto. Ou seja, caso se deseje utilizar um novo sistema hipertexto, basta desenvolver o segundo módulo, que possui os arquivos da análise do código fonte como entrada para que este gere os nós do hiperdocumento de acordo com as características do sistema hipertexto em questão.

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

Vale ressaltar que é fundamental que todos os módulos tenham sido compilados corretamente para que o hiperdocumento possua informações consistentes e fidedignas.

Não é obrigatório a edição do programa em algum formato pré-definido mas, para melhor utilização das funções do ambiente, é necessário que cada módulo possua um cabeçalho, em forma de comentário, contendo informações que documentem o módulo. Por exemplo, descrição do módulo (função e objetivo), algoritmo, autor, data da criação, etc.... Estas não precisam seguir nenhuma ordem, apenas precisam estar delimitadas por uma linha de comentário com mais de 10 caracteres '-' no início e outra no final do conjunto de informações

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 deverá criar para cada nó do tipo Módulo, a partir da análise do código, 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 deverá dar opção ao usuário 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 Complexidade*** e nó do tipo ***Avaliação 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 Operando e os nós 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 5.1)

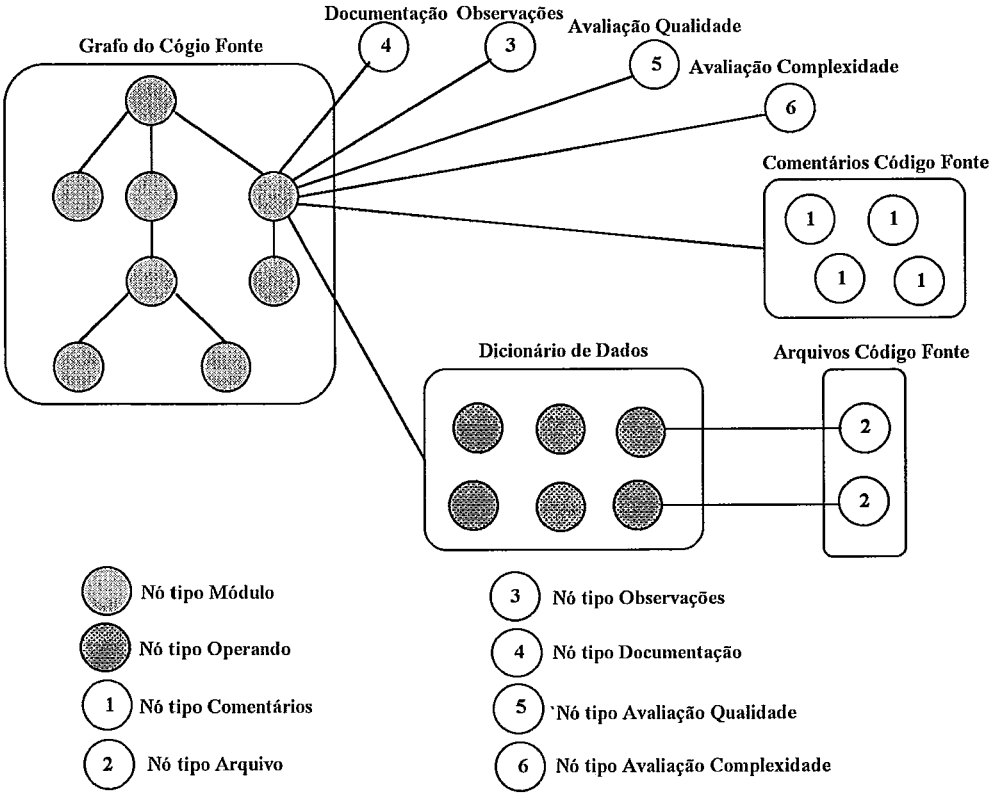


Figura 5.1. Esquema simplificado dos tipos de nós, contextos e ligações do HyFor

Conteúdo de cada tipo de nó criado automaticamente pelo ACF:

- **nó tipo Módulo**

Código fonte de um módulo com as linhas numeradas, sem os comentários entre as linhas. Contém apenas uma marcação na linha inicial do comentário que será utilizada como botão para o nó do tipo comentário correspondente. Todas as informações deste tipo de nó são geradas automaticamente e não poderão ser alteradas manualmente no sistema hipertexto.

- **nó tipo Operando**

O ACF gera este tipo de nó, automaticamente, para três tipos de operandos: variável, common e arquivo. Independente do tipo de operando, este tipo de nó sempre será gerado te em duas partes: uma contendo a descrição do operando, que inicialmente estará vazia e será preenchida pelo usuário, se desejado, no módulo autoria do sistema hipertexto; outra contendo informações obtidas pelo ACF durante a análise do código. O ACF gera este nó em duas partes visando o controle de alterações nos nós.

Dependendo do operando, a segunda parte do nó conterá informações diferentes.

Para o operando variável o conteúdo do nó será o seguinte: nome da variável, nome do módulo, tipo da variável, declarada (sim ou não), escopo (local ou global), forma de escopo global (common ou parâmetro), número de operações de entrada/saída, número de ocorrências, linha da primeira referência, linha da última referência, se for global do tipo parâmetro, lista todos os módulos para os quais a variável vai e dos quais ela vem, e os seus sinônimos nestes módulos.

Para o operando common o conteúdo do nó será o seguinte: nome do common e um quadro contendo todos os módulos que utilizam este common com suas respectivas declarações de listas de variáveis.

E para o operando arquivo, o conteúdo do nó será o seguinte: nome do arquivo, módulo que foi declarado, formato e status.

A primeira parte deste tipo de nó será criada e poderá ser alterada manualmente no sistema hipertexto, sempre que desejado. A segunda parte é gerada automaticamente e não poderá ser alterada manualmente.

- **nó tipo Arquivo**

Conteúdo dos arquivos utilizados pelo programa. É gerado sempre que existir um nó do tipo Operando contendo informações de um arquivo e este arquivo estiver disponível no esquema do hiperdocumento.

Seu conteúdo não poderá ser alterado manualmente no sistema hipertexto.

- **nó tipo Comentário**

Comentários referentes a um determinado conjunto de comandos do código fonte.

Todas as informações deste nó são geradas automaticamente pelo ACF e poderão ser alteradas manualmente no sistema hipertexto.

- **nó tipo Observações**

Este nó é gerado automaticamente sem conteúdo inicialmente. Deverá conter qualquer informação que explicite ou especifique o módulo que estará ligado a ele.

Deverá ser preenchido no módulo autoria do sistema hipertexto. Haverá a possibilidade das informações deste tipo de nó serem importadas de um arquivo externo podendo, desta forma, possuir diversos tipos de informação. Por exemplo, um gráfico que explicite o algoritmo implementado em um módulo poderá ser "scaniado" e importado para este nó, ficando disponível para navegação no sistema hipertexto, permitindo um melhor entendimento do módulo.

- **nó tipo Documentação**

Contém as informações extraídas automaticamente do cabeçalho do módulo, caso estas tenham sido editadas no código, de acordo com o formato citado anteriormente.

Estas informações poderão ser alteradas manualmente no sistema hipertexto.

- **nó tipo Avaliação Complexidade**

Contém o relatório de avaliação da complexidade do código, gerado pelo Avaliador de Complexidade do ACF. Este nó e suas ligações são geradas

automaticamente e não poderá ser alterado manualmente no sistema hipertexto.

- **nó tipo Avaliação Qualidade**

Contém o relatório de avaliação da qualidade do código, gerado pelo Avaliador de Qualidade do ACF. Este nó e suas ligações são geradas automaticamente e não poderá ser alterado manualmente no sistema hipertexto.

Ligações entre os nós geradas automaticamente pelo ACF:

O ACF gera 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. Abaixo são apresentadas todas as possíveis ligações entre os diversos tipos de nós gerados.

- **Ligações geradas para o nó tipo Módulo**

Cada nó do tipo Módulo estará ligado aos nós dos módulos que chama, montando o grafo de chamadas do programa. Será possível acessar, a partir do módulo chamador, o nó do tipo Módulo que contém o código fonte ou apenas o nó do tipo Documentação do módulo chamado. Estas duas ligações serão realizadas sobre um único botão marcado no nó do módulo chamador, utilizando-se um nó intermediário para direcionamento da ligação. O botão será o nome 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.

Também estará ligado aos nós do tipo Comentário, que se relacionam com ele, tendo como botão para este nó, um conjunto de caracteres do tipo "com__", onde "__" é um número sequencial. Este número vai sendo incrementado de acordo com a criação de mais um nó do tipo Comentário para o nó do tipo Módulo em questão. Este botão fica posicionado na primeira linha que continha o comentário extraído.

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 o nó 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. O botão será o nome do módulo.

Os nós do tipo Avaliação Complexidade e Avaliação Qualidade só serão ligados ao nó do tipo módulo que contém o código fonte do módulo principal do programa. O botão de ligação será o nome do módulo principal do programa. desta forma podemos observar que quando o nó tipo Módulo contiver o módulo principal, deverá ser utilizado um outro nó intermediário. Este nó direcionará para as ligações com um ou outro destes relatórios, para a ligação com o nó tipo Documentação e para a ligação com o nó tipo Observações.

A figura 5.2 mostra uma representação gráfica de todas as ligações geradas automaticamente pelo ACF, para este tipo de nó.

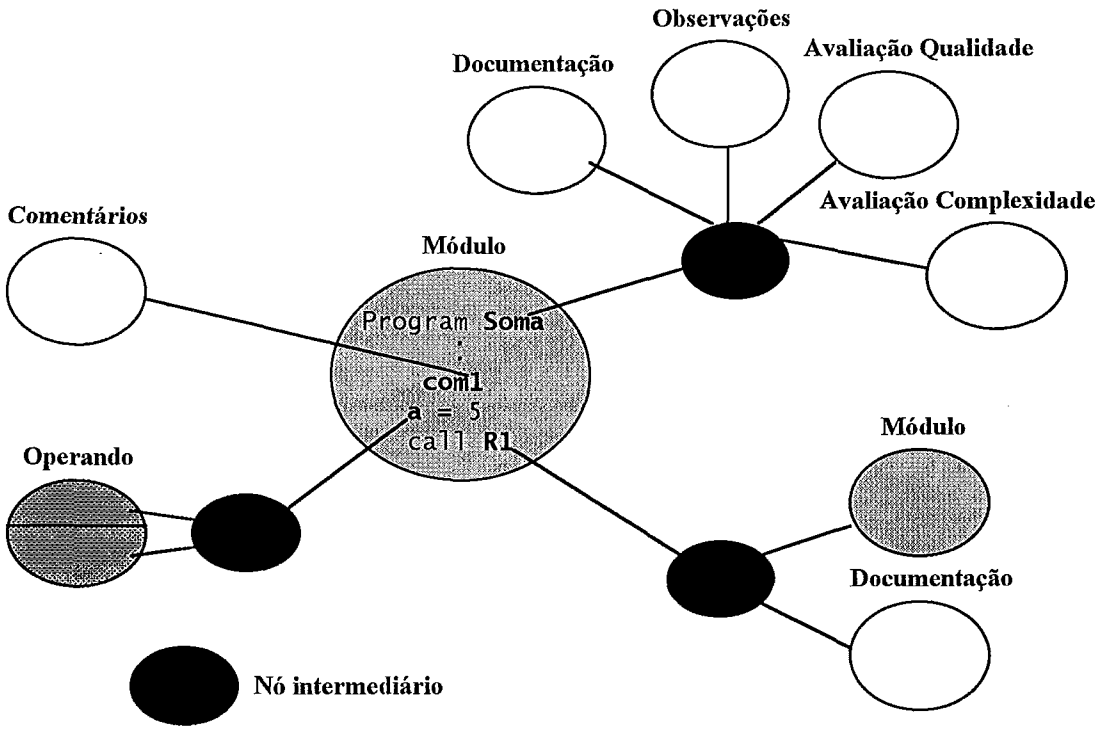


Figura 5.2. Esquematização das ligações do nó tipo Módulo

- **Ligações geradas para o nó 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 tipo Arquivo correspondente. Esta ligação só será feita automaticamente se o ACF conseguir criar o nó 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.

A figura 5.3 mostra uma representação gráfica de todas as ligações geradas automaticamente pelo ACF, para este tipo de nó.

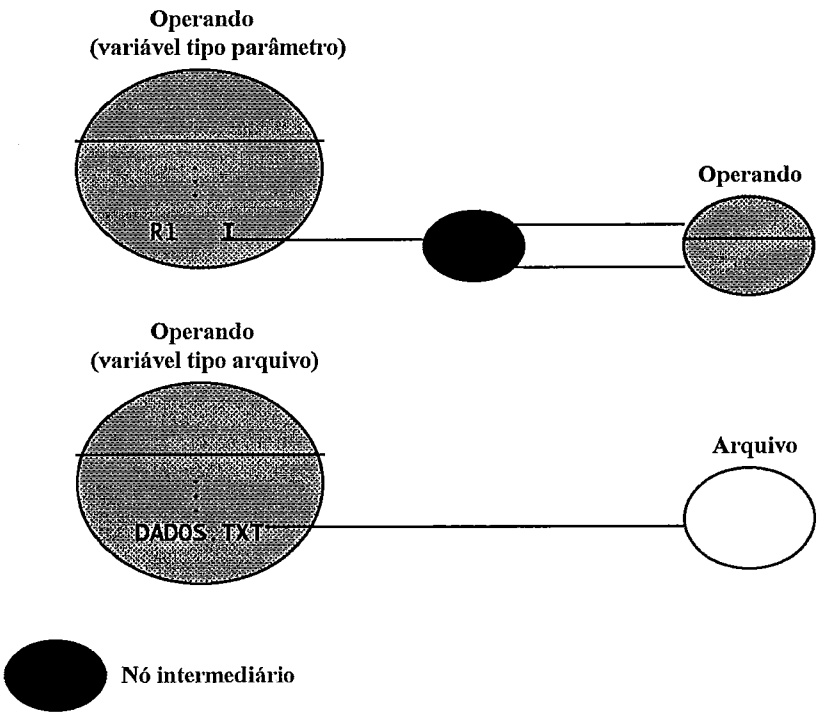


Figura 5.3. Esquematização das ligações do nó tipo Operando

- **Ligações geradas para o nó tipo Avaliação 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 tipo Avaliação 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 desejar não destruir os nós e ligações do hiperdocumento corrente, que foram criados e/ou alterados pelo usuário do hipertexto.

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 e depois entrará normalmente no hiperdocumento desejado. Vale ressaltar que as alterações, antes da regeração, não afetarão a versão corrente do hiperdocumento.

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.

Foram consideradas estas duas alternativas 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.

Chamamos de esquema a área que contém toda a base de dados do hiperdocumento. Sempre que uma nova versão do hiperdocumento for gerada automaticamente, será criado um subesquema do esquema corrente. Por exemplo, se o esquema é TESTE no diretório C:\HYFOR, a base de dados estará armazenada no diretório C:\HYFOR\TESTE. Quando uma nova versão for gerada, será criado o subesquema VERSAO_1 e a base de dados estará armazenada no diretório C:\HYFOR\TESTE\VERSAO_1.

Esta função permitirá a criação de dois novos tipos de nós: tipo ***Versão*** e tipo ***Alteração***.

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

O nó Versão conterá a versão anterior do código fonte e também estará ligado ao nó 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 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. Serão controladas até cinco versões de um módulo.

A figura 5.4 apresenta uma esquematização das ligações do nó tipo Módulo incluindo estes dois outros tipos de nós.

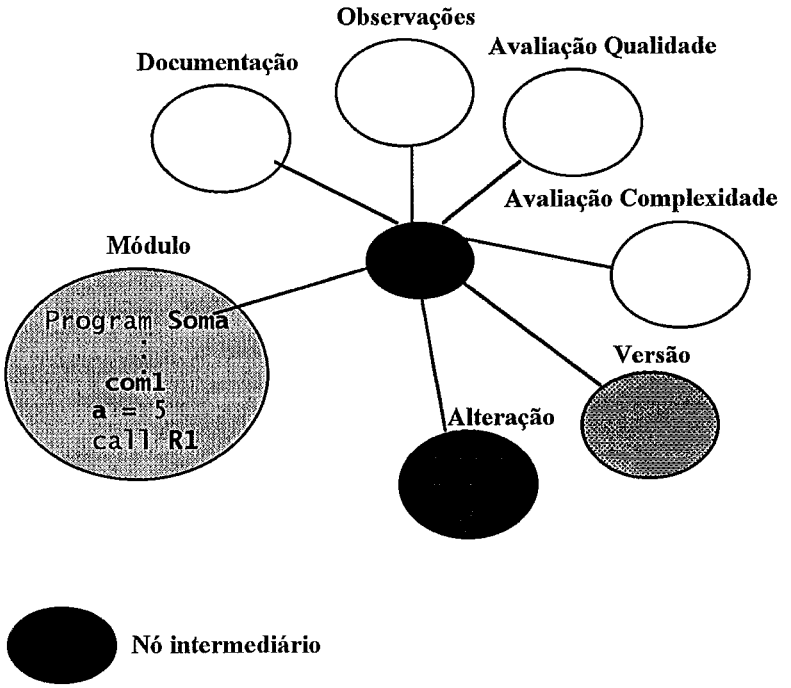


Figura 5.4. Esquematização das ligações dos nós tipo Versão e Alteração com o nó tipo Módulo

V.2.2. 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 nesta tese utilizou o sistema hipertexto ***Hiperbase*** e o

módulo de geração/regeração de hiperdocumento do ACF foi desenvolvido levando em conta as características básicas deste sistema. No próximo capítulo veremos maiores detalhes sobre a implementação do protótipo.

V.2.3. GAD

Esta ferramenta é responsável pela geração automática de documentação sobre o programa. 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.2.3.1. Especificação do programa

Conterá todas as informações sobre um programa possíveis de serem extraídas do hiperdocumento. São elas:

I. Identificação do programa
  I.1. Nome do programa
  I.2. Autor do hiperdocumento
  I.3. Data da criação do hiperdocumento

II. Grafo de chamadas dos módulos

III. Descrição de cada módulo
  III.1. Nome do módulo
  III.2. Descrição geral
  III.3. Módulos que chama
  III.4. Módulos pelos quais é chamado
  III.5. Dicionário de Dados
  III.6. Observações sobre a implementação

Os itens I.1, I.2 e I.3 são extraídos diretamente do arquivo de controle do hiperdocumento.

O item II é a representação gráfica do grafo de nós do tipo Módulo gerado pelo ACF.

O item III.1 é obtido do nó tipo Módulo.

O item III.2 é extraído do nó tipo Documentação ligado ao nó tipo Módulo em questão.

O itens III.3 e III.4 são extraídos do grafo de nós gerado pelo ACF.

O item III.5 é extraído dos nós tipo Operando ligados ao nó tipo Módulo em questão.

O item III.6 é obtido do nó tipo Observações ligado ao nó tipo Módulo em questão.

V.2.3.2. Especificação do módulo

Conterá todas as informações sobre um módulo, possíveis de serem extraídas do hiperdocumento. São as mesmas listadas no item III da Especificação do programa.

V.2.3.3. Relatório de controle de alterações do programa

Conterá todas as informações sobre alterações, em relação a versão anterior, realizadas em cada um dos módulos de um programa. Serão listados todos os módulos que compõem um programa, mesmo que não posssua versão anterior.

As informações listadas serão as seguintes:

I. Identificação do programa
- I.1. Nome do programa
- I.2. Autor do hiperdocumento
- I.3. Data da criação do hiperdocumento

II. Identificação de cada módulo
- II.1. Nome do módulo
- II.2. Descrição geral
- II.3. Alterações em relação a versão anterior
- II.4. Autor da documentação das alterações
- II.5. Data da documentação das alterações

Os itens I.1, I.2 e I.3 são extraídos diretamente do arquivo de controle do hiperdocumento.

O item II.1 é obtido do nó tipo Módulo.

O item II.2 é extraído do nó tipo Documentação ligado ao nó tipo Módulo em questão.

O itens II.3, II.4 e II.5 são extraídos do nó tipo Alteração ligado ao nó tipo Módulo em questão.

V.2.3.4. Relatório de controle de alterações de um módulo

Conterá todas as informações sobre alterações, em relação a versão anterior, realizadas em um módulo do programa. São as mesmas listadas no item II do Relatório de controle de alterações do programa.

V.3. Conclusão preliminar

Este capítulo descreveu os objetivos deste trabalho, mostrando os problemas enfrentados no desenvolvimento de software científico, utilizando conceitos descritos nos capítulos anteriores, e como estes problemas podem ser amenizados com a utilização de um ambiente CASE chamado HyFor.

Os últimos itens deste capítulo compõem uma especificação informal do HyFor.

O próximo capítulo descreverá o protótipo do HyFor desenvolvido a partir desta especificação e apresentará um exemplo de utilização do ambiente.

VI - Protótipo

Este capítulo descreve o protótipo desenvolvido para o HyFor e apresenta um exemplo de utilização do ambiente.

VI.1. 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.

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

A exigência do tamanho de memória é do Analisador de Código Fortran (ACF).

Abaixo será feita a descrição da implementação de cada uma das ferramentas que compõem o HyFor.

VI.1.1. 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, com

exceção do módulo de reestruturação que foi o único não implementado no protótipo.

Na figura 5.1 apresentamos uma visão esquemática dos módulos do ACF e seus interrelacionamentos.

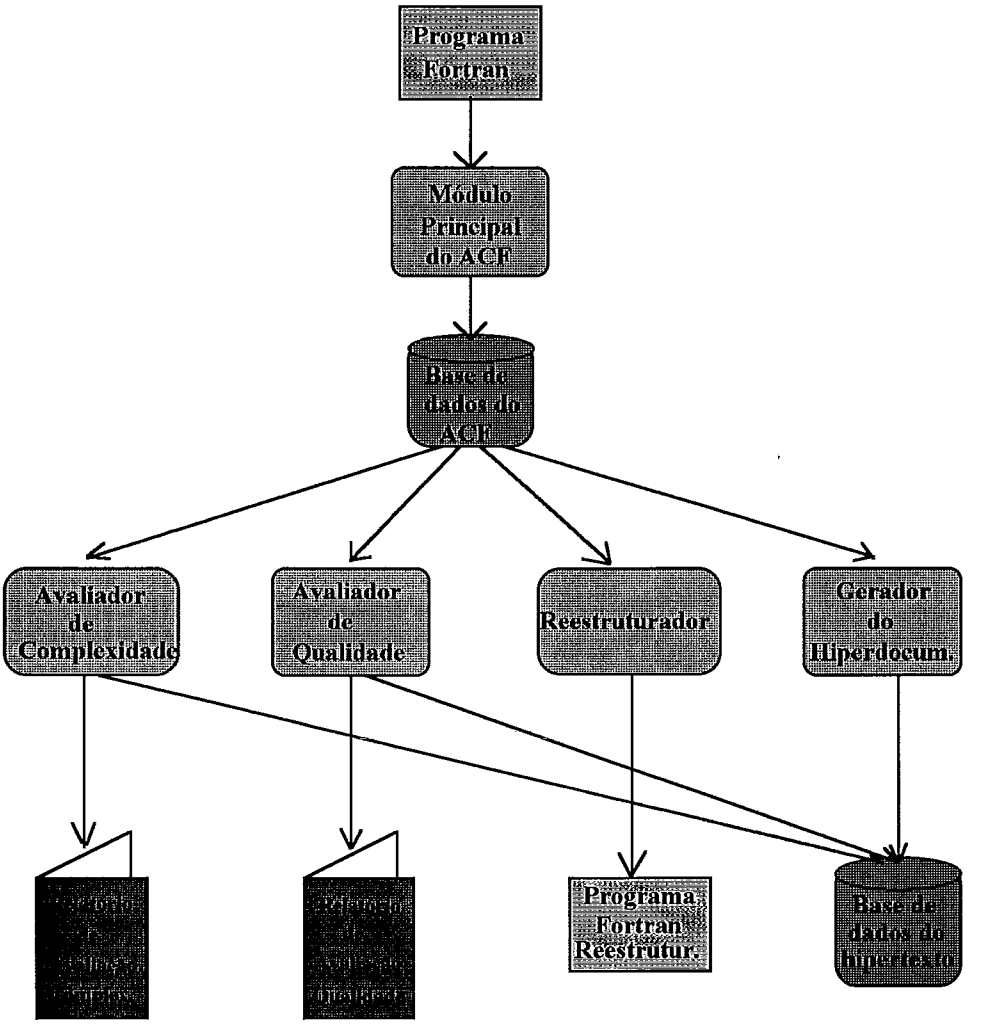


Figura 6.1. Visão esquemática do ACF

VI.1.2. 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 [RIBE92] [CERQ93] [BENT93]. 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.

A figura abaixo apresenta uma tabela que descreve as principais características de sistemas hipertexto, incluindo o Hiperbase.

	Estruturas Hierárquicas	Ligações Referenciais	Nós e Lig. Tipadas	Versões	Procura por string ou chave	Multi-usuário	Suporta Gráficos	Navegador Gráfico
Intermedia	Sim	Sim	Sim	Não	Sim	Sim	Sim	Sim
KMS	Múltiplas	Sim	Não	Sim	Sim	Sim	Sim	Não
Neptune	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
NLS/Augment	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Não
NoteCards	Múltiplas	Sim	Apenas nós	Não	Sim	Sim	Sim	Sim
Textnet	Múltiplas	Sim	Sim	Não	Apenas chave	Não	Não	Não
Hyperties	Não	Sim	Não	Não	Não	Não	Sim	Não
Xanadu	Não	Sim	Sim	Sim	. Não	Não	Sim	Não
ZOG	Sim	Não	Não	Não	Por texto	Sim	Não	Não
HIPERBASE	Não	Sim	`Apenas nós	Não	Não	Não	Sim	Não

Figura 6.2. Principais características de alguns sistemas hipertexto

VI.1.3. 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.

O apêndice D apresenta um exemplo de cada um dos quatro primeiros relatórios e os apêndices B e C apresentam exemplos dos dois últimos, respectivamente.

VI.1.4. Considerações gerais

VI.1.4.1. Módulo de utilitários

Foi necessário acrescentar no HyFor um módulo de utilitários de programação Fortran, visando a maior facilidade de utilização do ambiente por parte do especialista.

Vale ressaltar que não é conhecido no mercado nenhum ambiente do tipo Turbo-C ou Turbo-Pascal, para a linguagem Fortran em microcomputadores, que integre estes utilitários.

A Borland iniciou o desenvolvimento de um ambiente deste tipo (parecido com o Turbo-C 1.0), mas parou o projeto sem comercializar o produto

Existem no mercado alguns editores inteligentes do tipo BRIEF e EPSYLON para DOS e o EMACS para UNIX, que permitem a montagem do ambiente integrado desejado, pelo próprio usuário. Mas estes não são amplamente utilizados em áreas

científicas.

A Microsoft utiliza o windows para tornar os utilitários do Fortran disponíveis para o usuário. Este produto já está no mercado mas não chega a ser um ambiente integrado e também não é utilizado em áreas científicas, de uma maneira geral.

Os utilitários que fazem parte deste módulo do HyFor são os seguintes: o compilador Fortran e o depurador Code View da Microsoft e o editor de textos SPF/PC.

É importante ressaltar que o HyFor simplesmente facilita a utilização destes utilitários através do ambiente, não interferindo no processamento destes.

VI.1.4.2. Módulo de definição

Todos os módulos do Hyfor trabalham sobre uma área de dados corrente. A definição desta área é feita dentro do ambiente, por um módulo especial de definição, e é obrigatória antes da utilização de qualquer outro módulo.

Esta definição inclui as seguintes informações: nome do hiperdocumento, diretório base do arquivo .FOR e nome do arquivo .FOR.

O módulo de definição do HyFor também recupera informações que serão utilizadas no sistema hipertexto, são elas: nome do usuário e senha.

A figura 6.3 apresenta a relação entre os módulos principais do HyFor e a área de dados corrente.

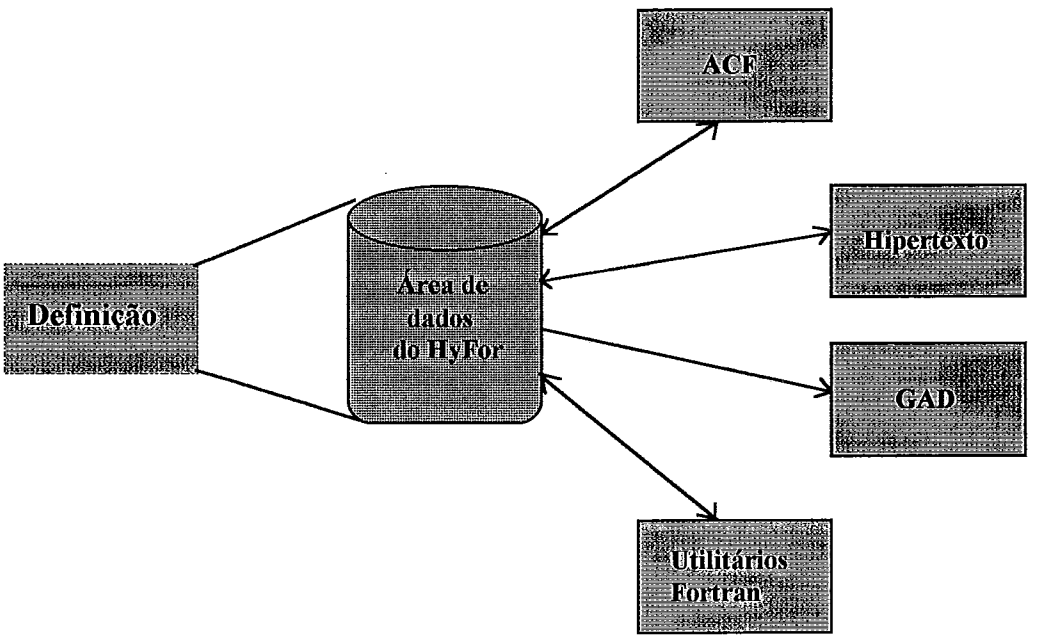


Figura 6.3. Visão esquemática da relação entre os módulos do HyFor e a área de dados corrente

VI.2. Exemplo de utilização

Para ilustrar o uso do HyFor foi utilizado um programa chamado MESH. Este programa foi desenvolvido pelo Setor de Desenvolvimento de Métodos do Centro de Pesquisa da Petrobrás.

O objetivo do programa é gerar malha de plataformas do tipo TLP ou semisubmersível com colunas circulares e pontoons retangulares.

Vale ressaltar que o código apresentado não será julgado quanto a sua forma de resolver o problema, apenas servirá de meio para mostrar a funcionalidade do ambiente proposto.

O apêndice E apresenta a listagem do código do programa MESH.

VI.2.1. Execução do HyFor

Após a execução do comando:

C:\HYFOR> **HyFor**

Será exibida a tela inicial mostrada na figura 6.4.

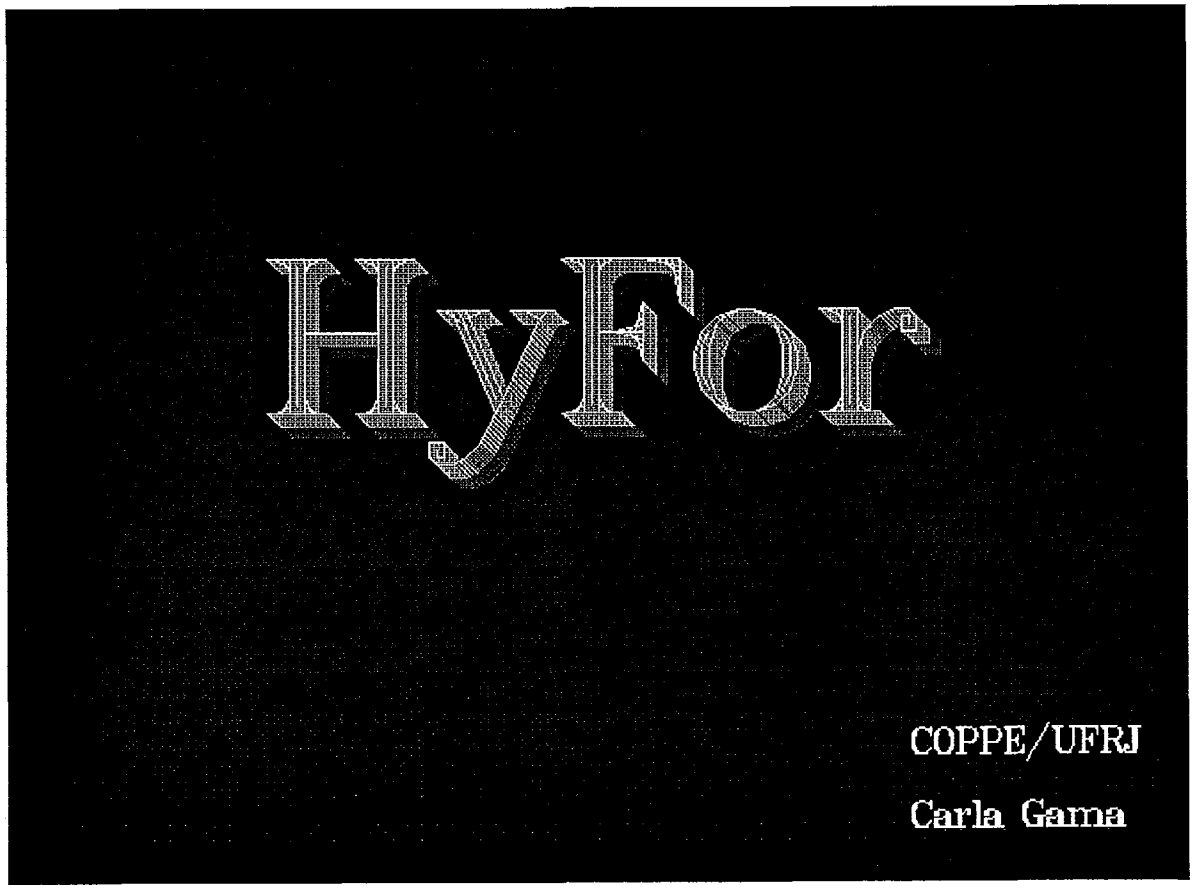


Figura 6.4. Tela inicial do HyFor

Após alguns segundos será exibida a tela com o menu principal do ambiente. A figura 6.5 ilustra esta tela.

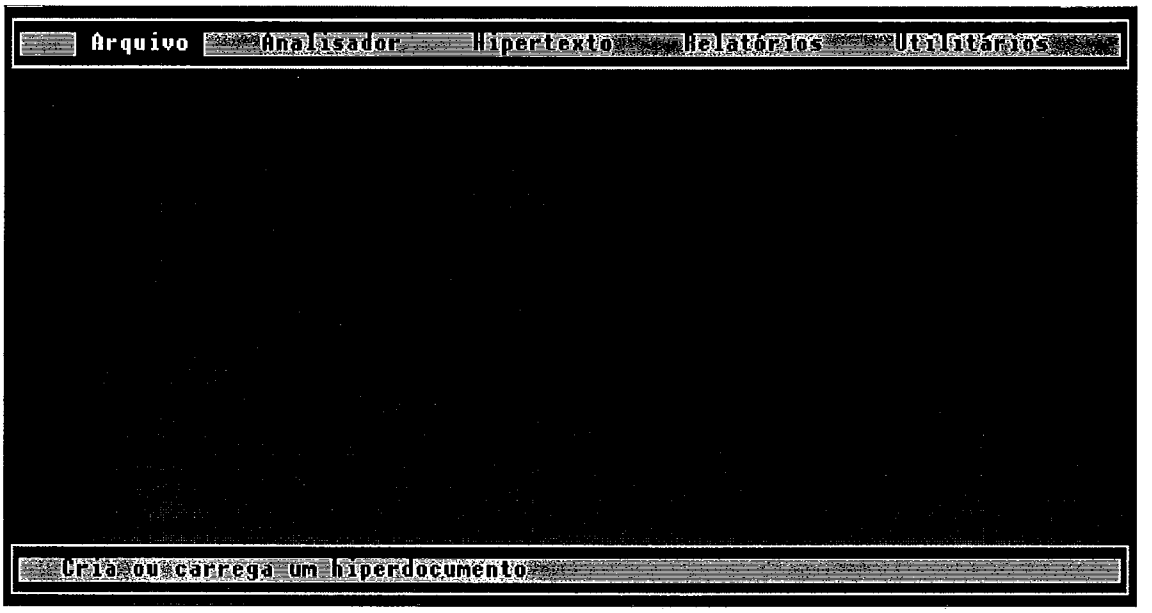


Figura 6.5. Menu principal do HyFor

O usuário deverá sempre executar a opção *Arquivo* para definir a área de dados corrente.

VI.2.1.1. Opção *Arquivo*

A execução desta opção equivale a apresentação de algumas subopções.

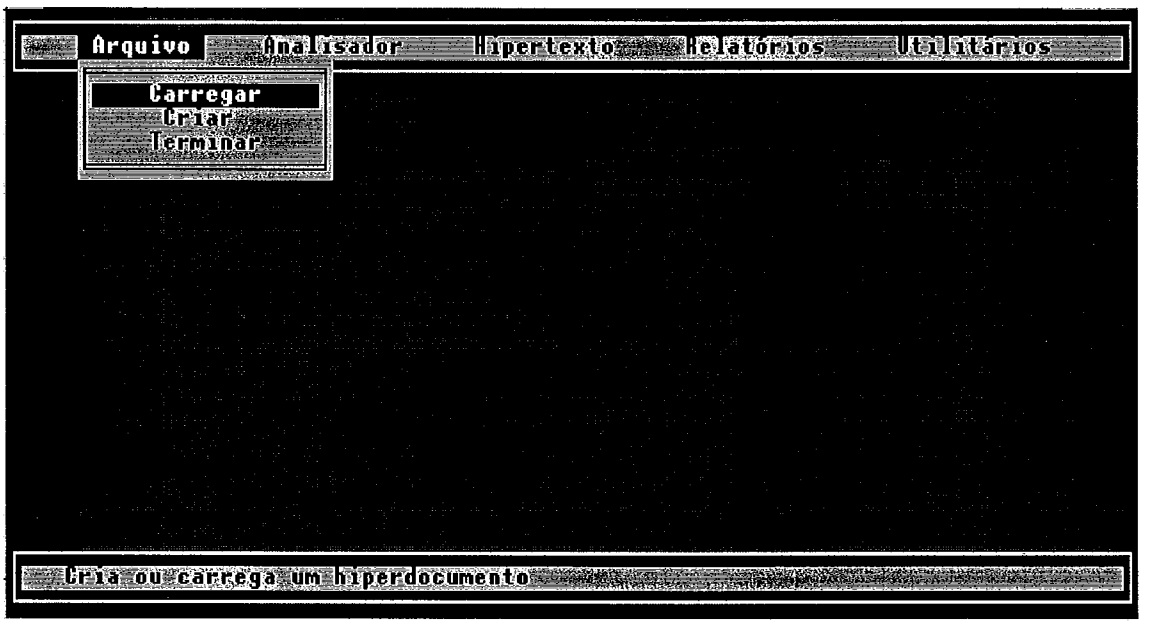


Figura 6.6. Opção *Arquivo*: Tela de subopções

VI.2.1.1.1. Subopção *Carregar*

Esta subopção é selecionada quando o usuário deseja utilizar um hiperdocumento já existente no ambiente. A execução desta subopção equivale a apresentação da tela mostrada na figura 6.7, onde o usuário deverá preencher as informações solicitadas.

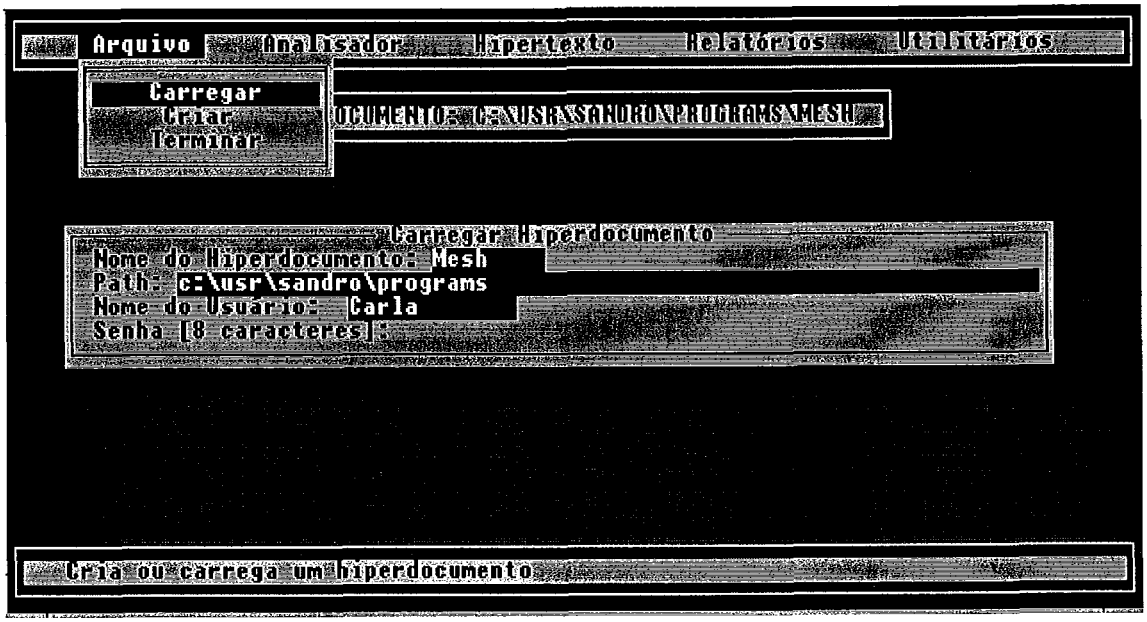


Figura 6.7. Subopção *Carregar*: Informações solicitadas para carregar um hiperdocumento já existente no ambiente

VI.2.1.1.2. Subopção *Criar*

Esta subopção é selecionada quando o usuário deseja criar um novo hiperdocumento. A execução desta subopção equivale a apresentação da tela mostrada na figura 6.8, onde o usuário deverá preencher as informações solicitadas.

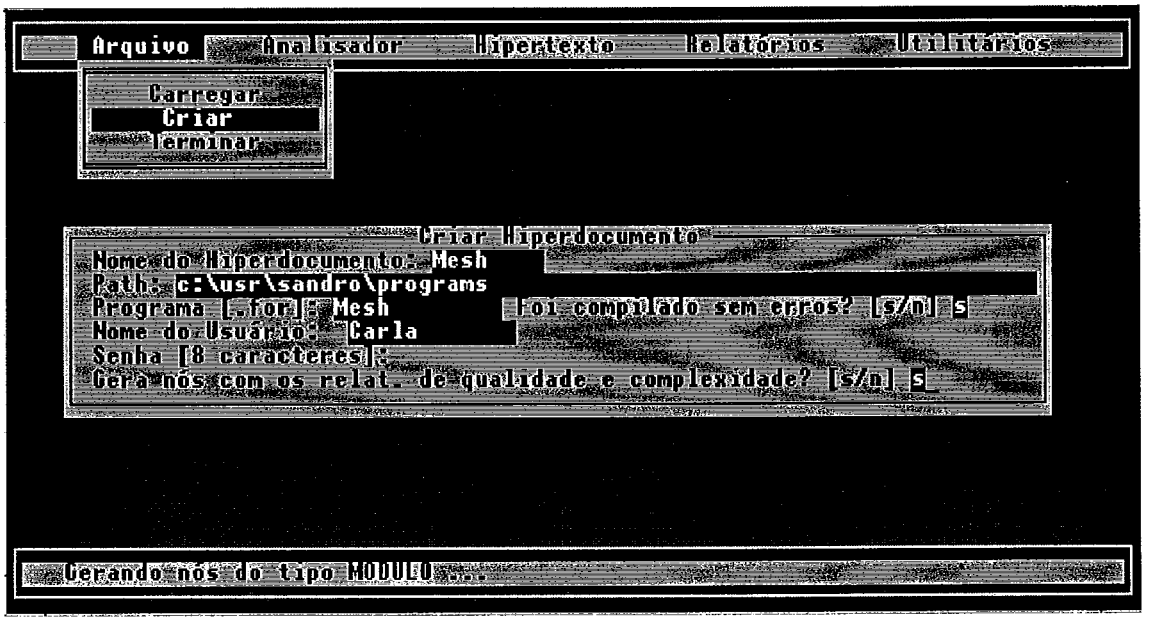


Figura 6.8. Subopção *Criar*: Informações solicitadas para criar um hiperdocumento

VI.2.1.1.3. Subopção *Terminar*

Esta subopção é selecionada quando o usuário deseja sair do HyFor.

VI.2.1.2. Opção *Analisador*

A execução desta opção equivale a apresentação de algumas subopções (ver figura 6.9).

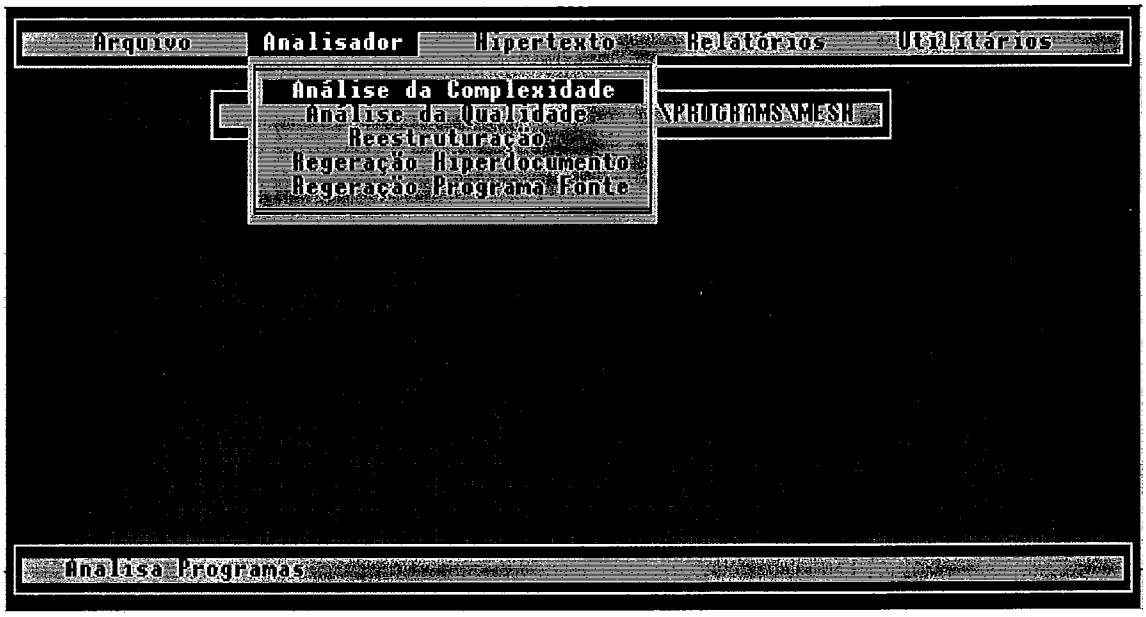


Figura 6.9. Opção *Analisador*: Tela de subopções

VI.2.1.2.1. Subopção *Análise da Complexidade*

Esta subopção é selecionada quando o usuário deseja gerar o Relatório de Avaliação da Complexidade do programa. Este relatório estará disponível para impressão na opção *Relatórios.*

É importante ressaltar que se este relatório for gerado nesta subopção independentemente da geração/regeração do hiperdocumento, ele não será ligado ao hiperdocumento.

O apêndice B apresenta o relatório de avaliação da complexidade gerado para o programa MESH.

VI.2.1.2.2. Subopção *Análise da Qualidade*

Esta subopção é selecionada quando o usuário deseja gerar o Relatório de Avaliação da qualidade do programa. Este relatório estará disponível para impressão na opção *Relatórios.*

É importante ressaltar que se este relatório for gerado nesta subopção independentemente da geração/regeração do hiperdocumento, ele também não será

ligado ao hiperdocumento.

O apêndice C apresenta o relatório de avaliação da qualidade gerado para o programa MESH.

VI.2.1.2.3. Subopção *Reestruturação*

Esta subopção é selecionada quando o usuário deseja realizar a reestruturação do código. Este módulo não foi implementado no protótipo.

VI.2.1.2.4. Subopção *Regeração Hiperdocumento*

Esta subopção é selecionada quando o usuário deseja regerar um hiperdocumento cujos nós do código fonte foram alterados.

A execução desta subopção equivale a apresentação das telas mostradas na figura 6.10 e 6.11, onde o usuário deverá preencher as informações solicitadas. A tela 6.11 só será apresentada se o usuário desejar criar nova versão do hiperdocumento e aparecerá para cada módulo alterado.

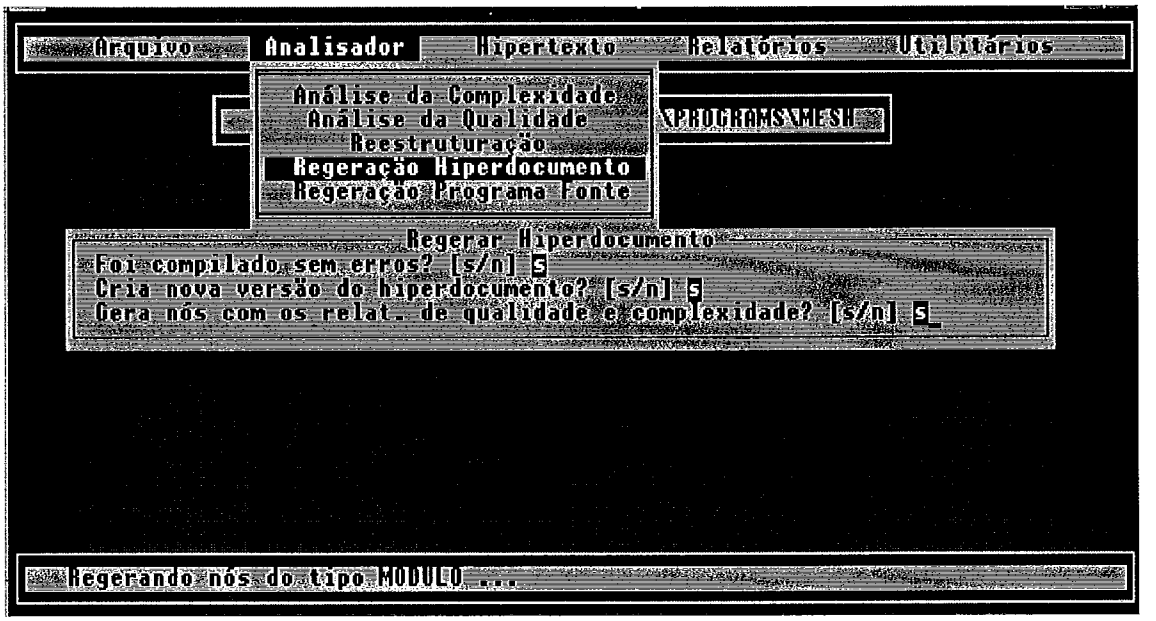


Figura 6.10. Subopção *Regeração Hiperdocumento:*Informações solicitadas para regerar um hiperdocumento

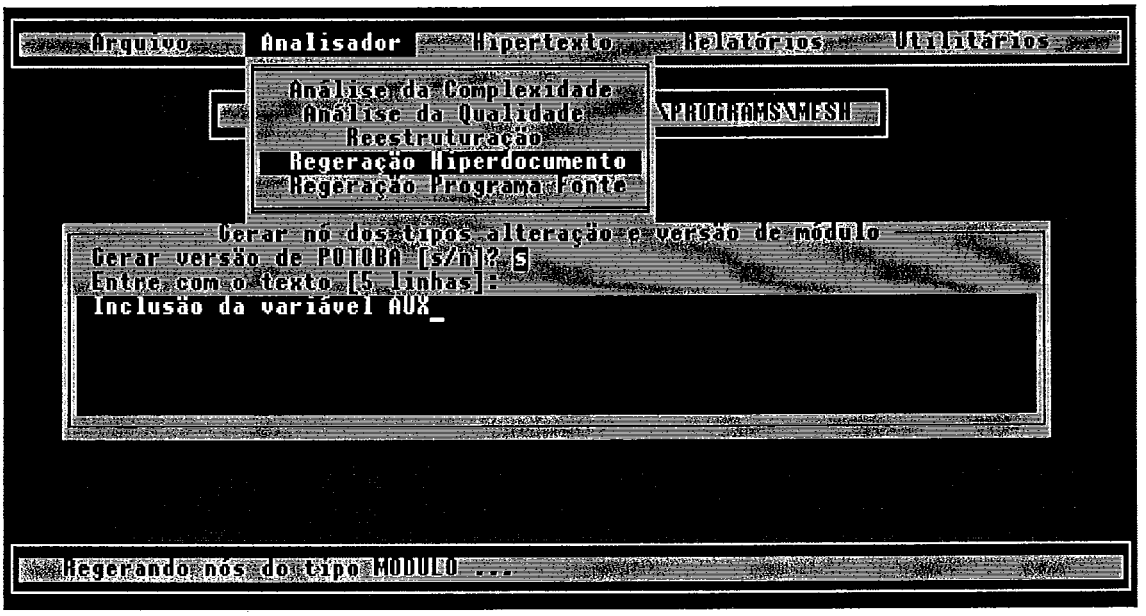


Figura 6.11. Subopção *Regeração Hiperdocumento:*Informações solicitadas para gerar nós do tipo alterações e versão para cada módulo

VI.2.1.2.5. Subopção *Regeração Programa Fonte*

Esta subopção é selecionada quando o usuário deseja regerar o programa fonte a partir dos nós do hiperdocumento.

Esta subopção é importante porque o usuário pode desejar montar o programa com as alterações que por ventura tenham sido efetuadas através do Hiperbase.

VI.2.1.3. Opção *Hipertexto*

A execução desta opção equivale a apresentação de algumas subopções (ver figura 6.12).

Figura 6.12. Opção *Hipertexto*: Tela de subopções

VI.2.1.3.1. Subopção *Autoria*

Esta subopção é selecionada quando o usuário deseja entrar no módulo Autoria do Hiperbase.

A figura 6.13 apresenta a tela de entrada do módulo Autoria do Hiperbase.

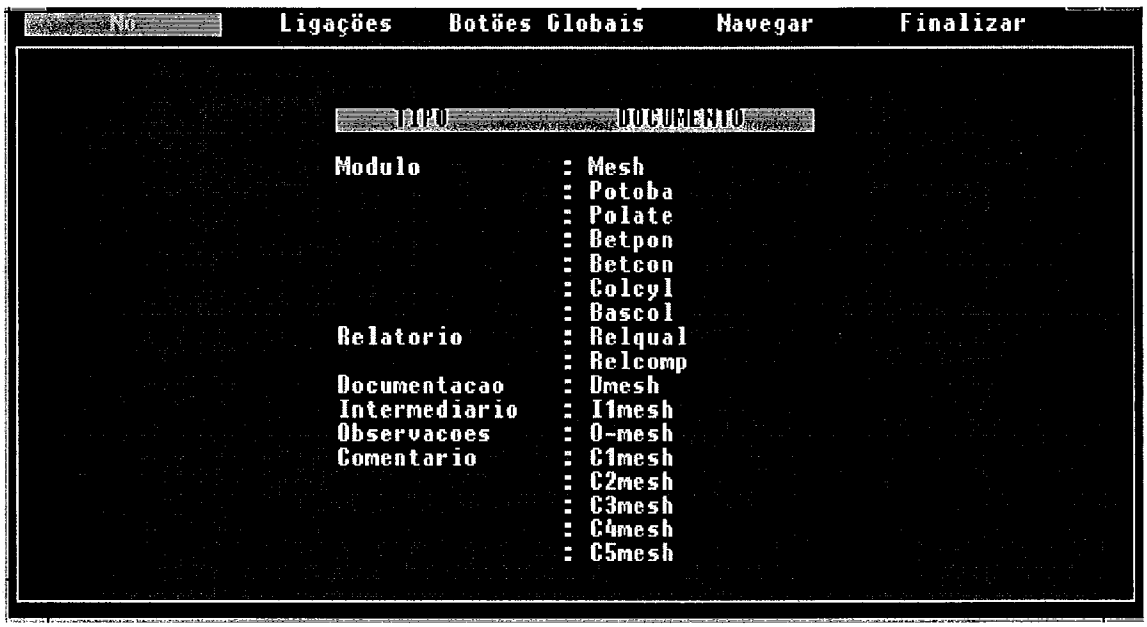


Figura 6.13. Subopção *Autoria*: Tela de entrada do Hiperbase/Autoria

Somente os nós do tipo Documentação, Comentários, Observações, Alterações e os nós criados pelo usuário podem ser alterados através do Hiperbase. O ACF prepara os outros tipos de nós para possuírem acesso negado neste módulo do Hiperbase.

No módulo Autoria o usuário pode criar quantos nós desejar e ligar estes aos nós já existentes do hiperdocumento, marcando os botões de ligação. Os nós criados sem tipo pré-definido pelo ACF durante a geração automática do hiperdocumento, não serão utilizados na geração automática dos documentos pelo GAD.

A figura 6.14 apresenta um nó do tipo Comentário do módulo MESH.

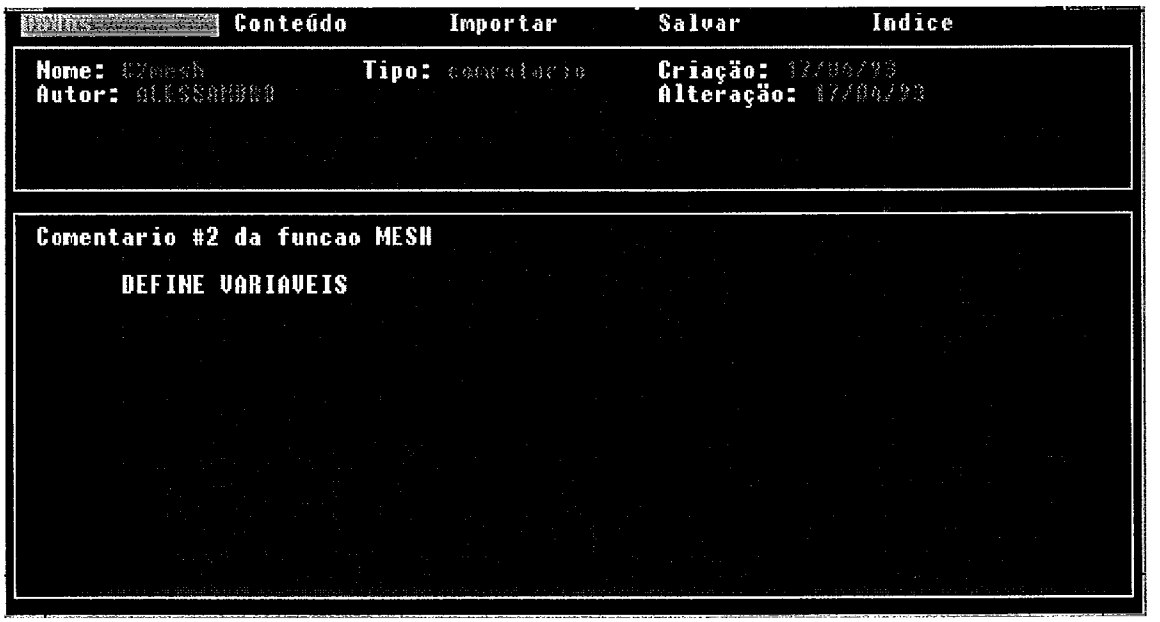


Figura 6.14. Subopção Autoria: Visão de um nó do tipo Comentário

VI.2.1.3.2. Subopção *Navegação*

Esta subopção é selecionada quando o usuário deseja entrar no módulo Navegação do Hiperbase.

A figura 6.15 apresenta a tela de entrada do módulo Navegação do Hiperbase.

Figura 6.15. Subopção *Navegação*: Tela de entrada do Hiperbase/Navegação

Neste módulo todos os tipos de nós são acessados e os botões de cada nó são apresentados em "highlight".

O objetivo inicial na criação do protótipo era sempre apresentar diretamente a tela contendo o nó do módulo principal do programa, não passando pelo índice dos nós. Mas, devido a uma característica do Hiperbase, não foi possível implementar esta idéia. O usuário sempre entrará primeiramente no índice e escolherá o nó inicial para consulta ao hiperdocumento.

Vale ressaltar que foi preciso criar uma padronização para os nomes dos nós gerados automaticamente a partir do código (ver apêndice F). Esta nomenclatura tenta ser o mais clara possível, mas consideramos que é preciso o usuário ter conhecimento dessas regras para consultar com mais facilidade o índice do hiperdocumento.

As figuras 6.16 a 6.25 apresentam uma sequência de consulta ao hiperdocumento do programa MESH. Esta consulta foi iniciada no nó do tipo Módulo contendo o módulo principal do programa MESH. As legendas das figuras informam o botão que foi acionado na tela anterior para acesso a tela em questão.

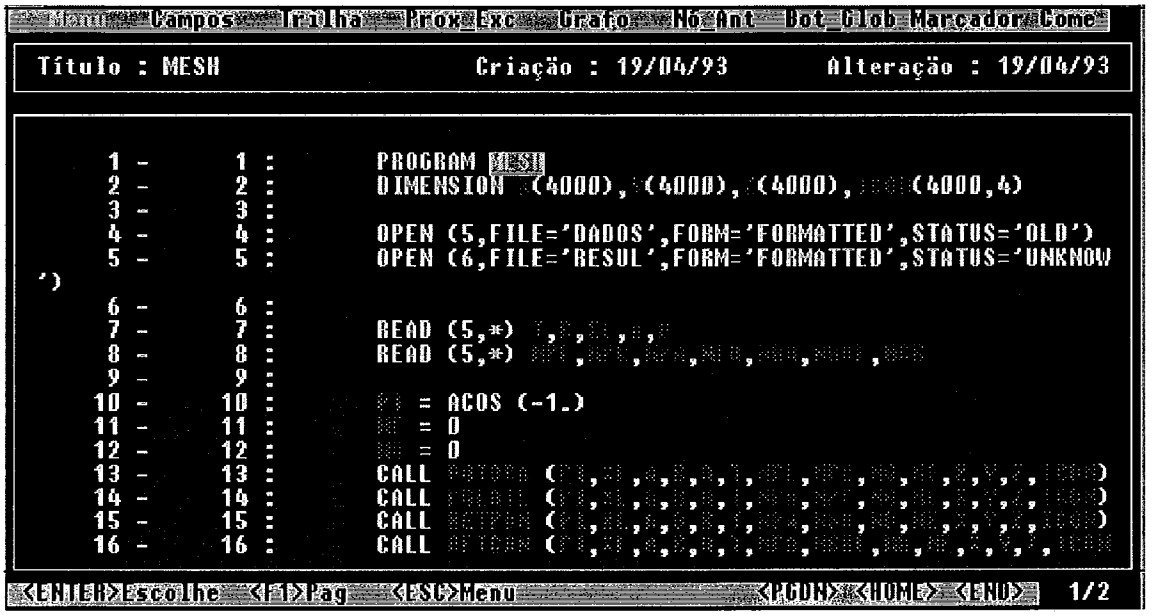


Figura 6.16. Navegação: Tela apresentada após seleção no índice do nó Mesh

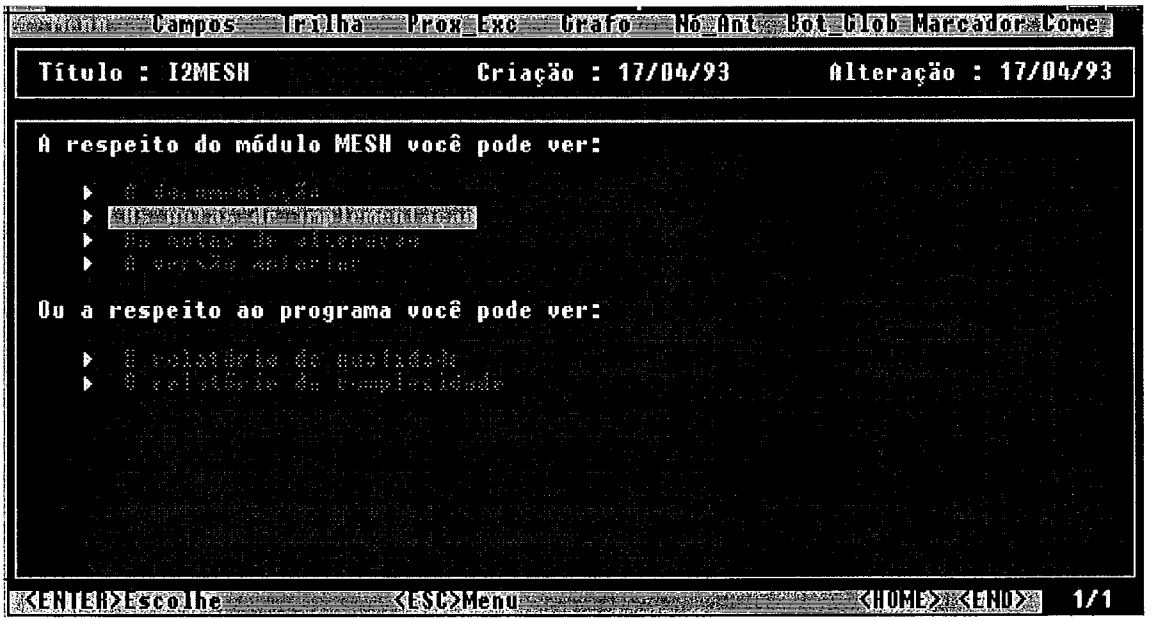


Figura 6.17. Navegação: Tela apresentada após seleção do botão *Mesh*

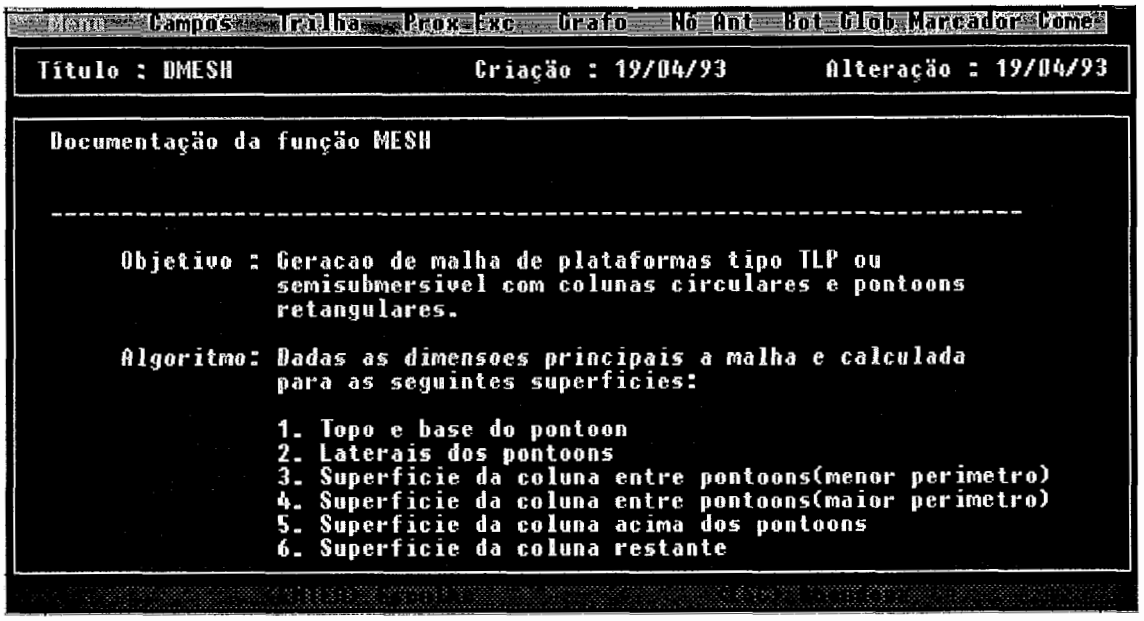


Figura 6.18. Navegação: Tela apresentada após seleção do botão *A Documentação*

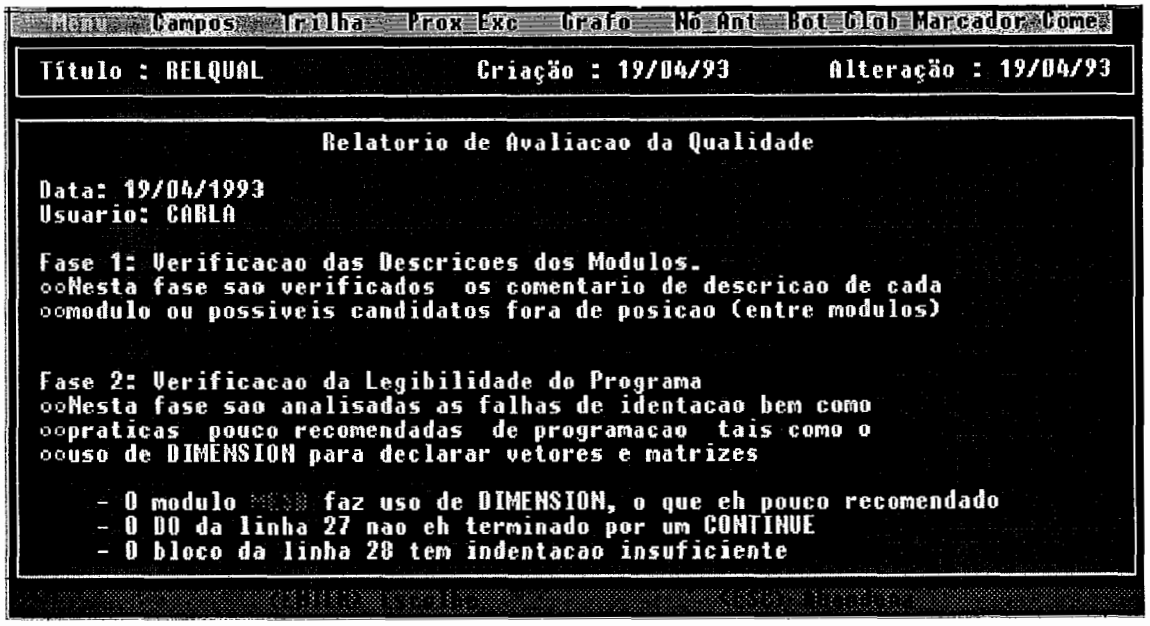


Figura 6.19. Navegação: Tela apresentada após retorno ao nó anterior e seleção do botão *o relatório de qualidade*

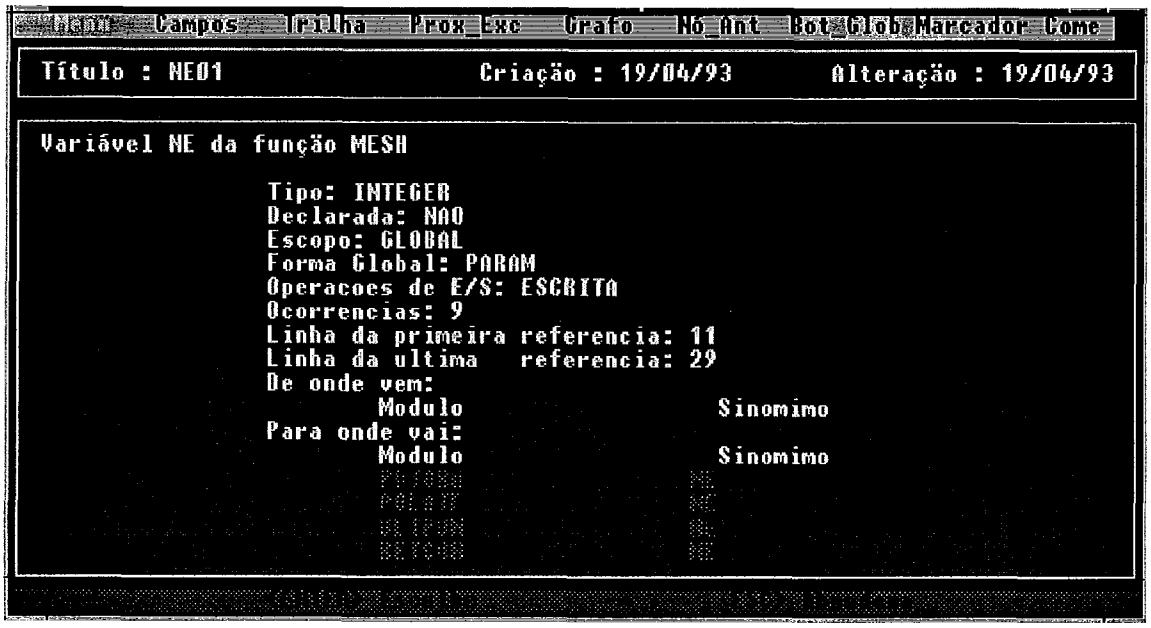


Figura 6.20. Navegação:Tela apresentada após retorno ao nó anterior que contém o módulo Mesh e seleção do botão *NE*

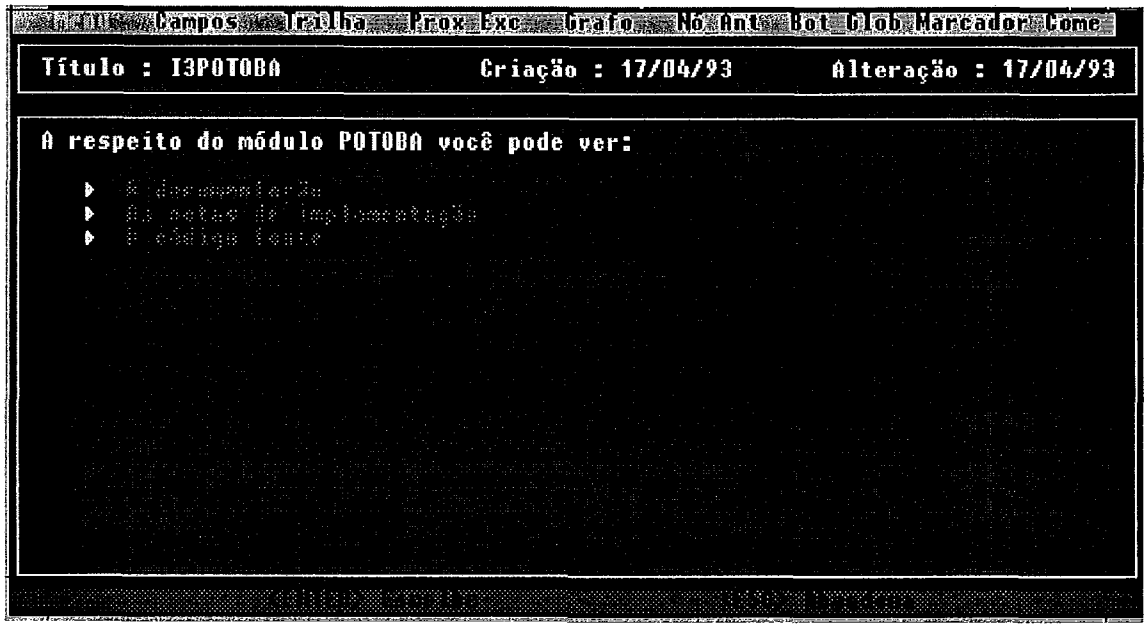


Figura 6.21. Navegação:Tela apresentada após retorno ao nó anterior que contém o módulo Mesh e seleção do botão *POTOBA*

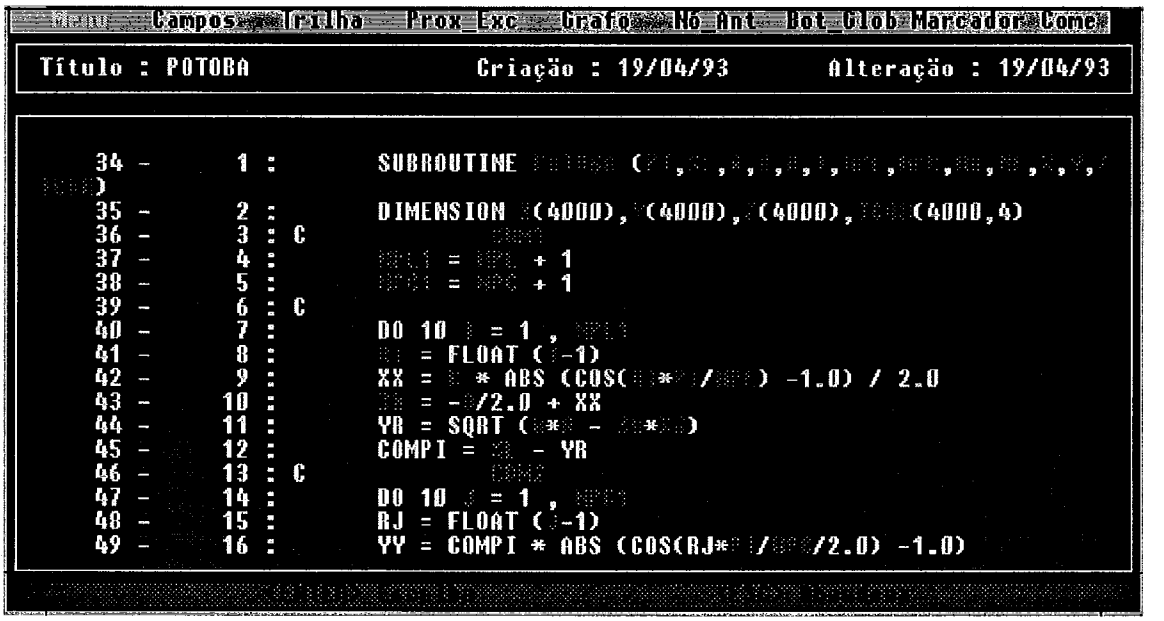


Figura 6.22. Navegação: Tela apresentada após seleção do botão *O código fonte*

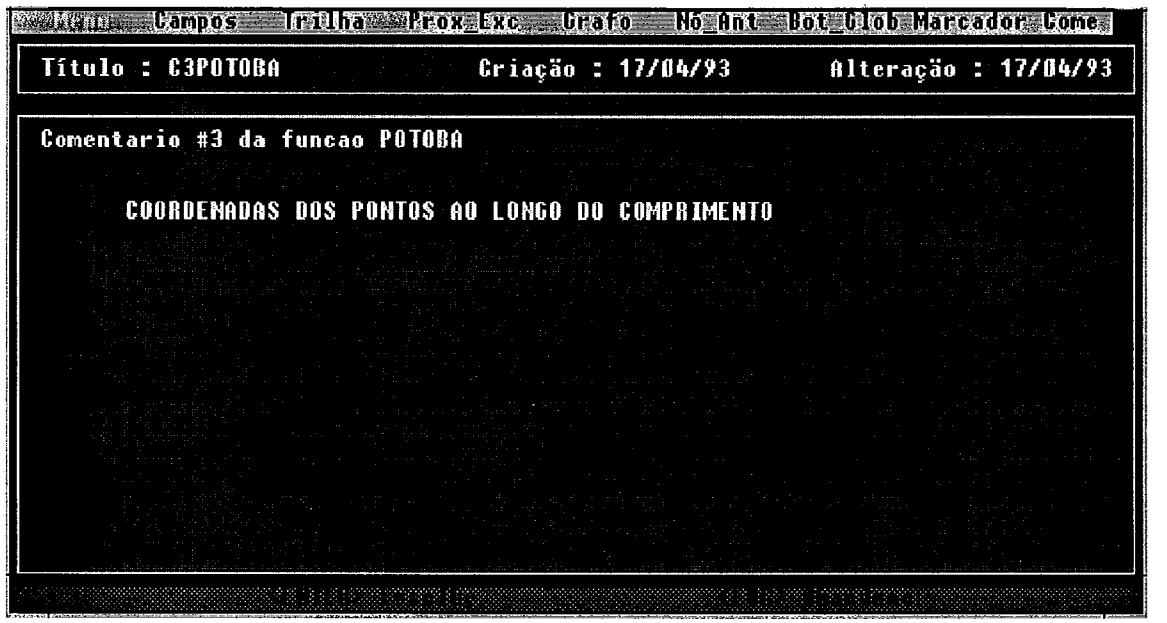


Figura 6.23. Navegação: Tela apresentada após seleção do botão *COM3*

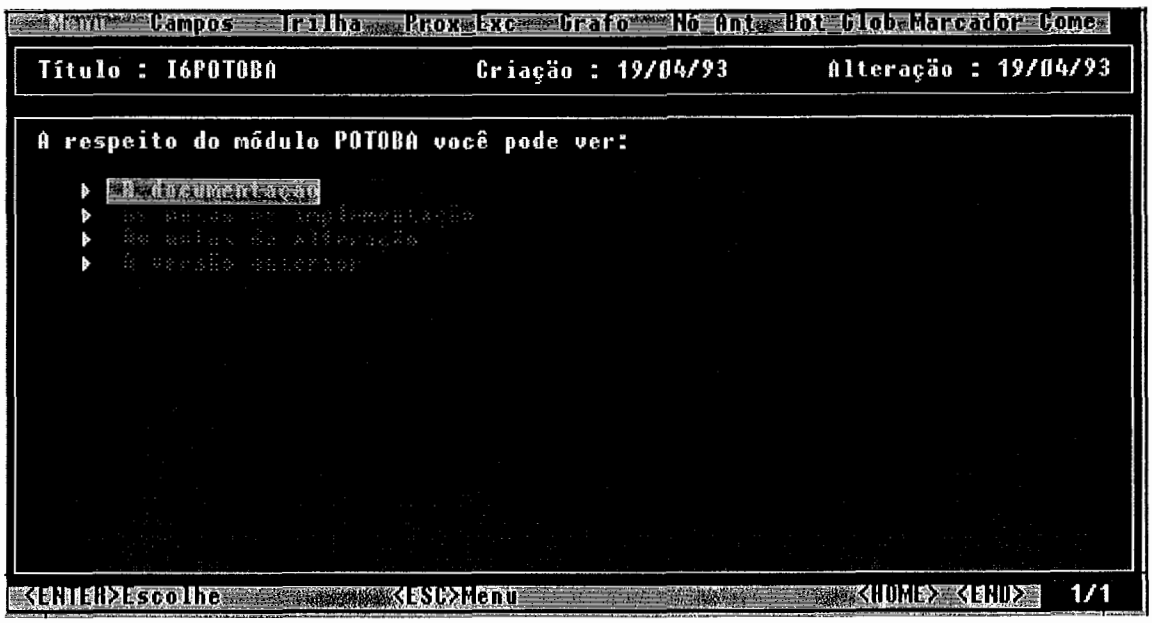


Figura 6.24. Navegação:Tela apresentada após retorno ao nó anterior que contém o módulo Potoba e seleção do botão *POTOBA*

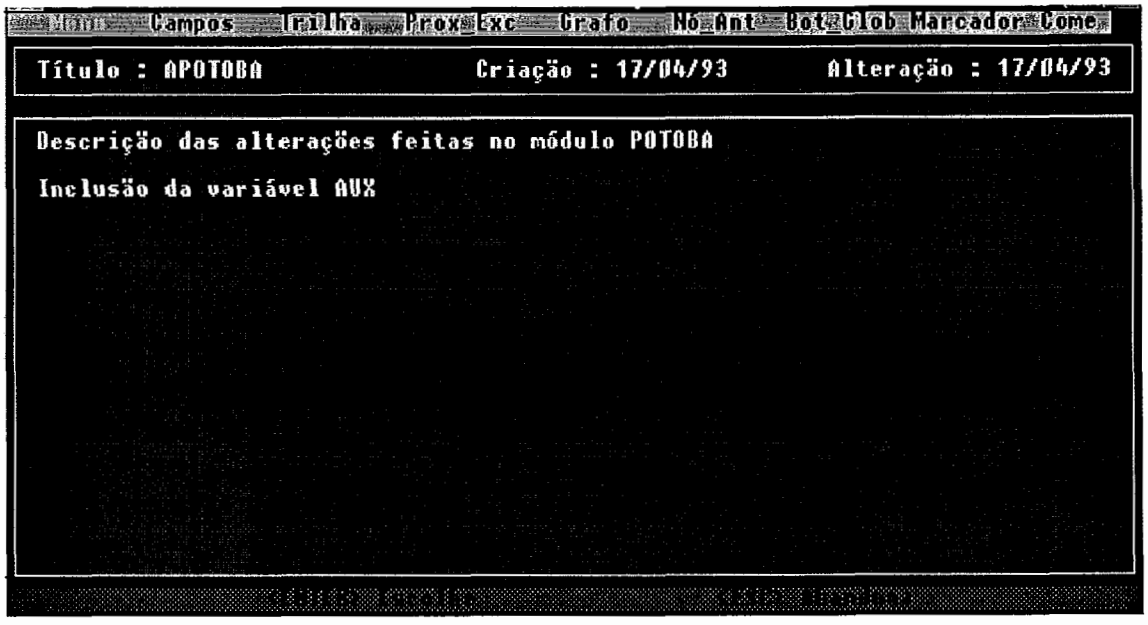


Figura 6.25. Navegação: Tela apresentada após seleção do botão *As notas de alteração*

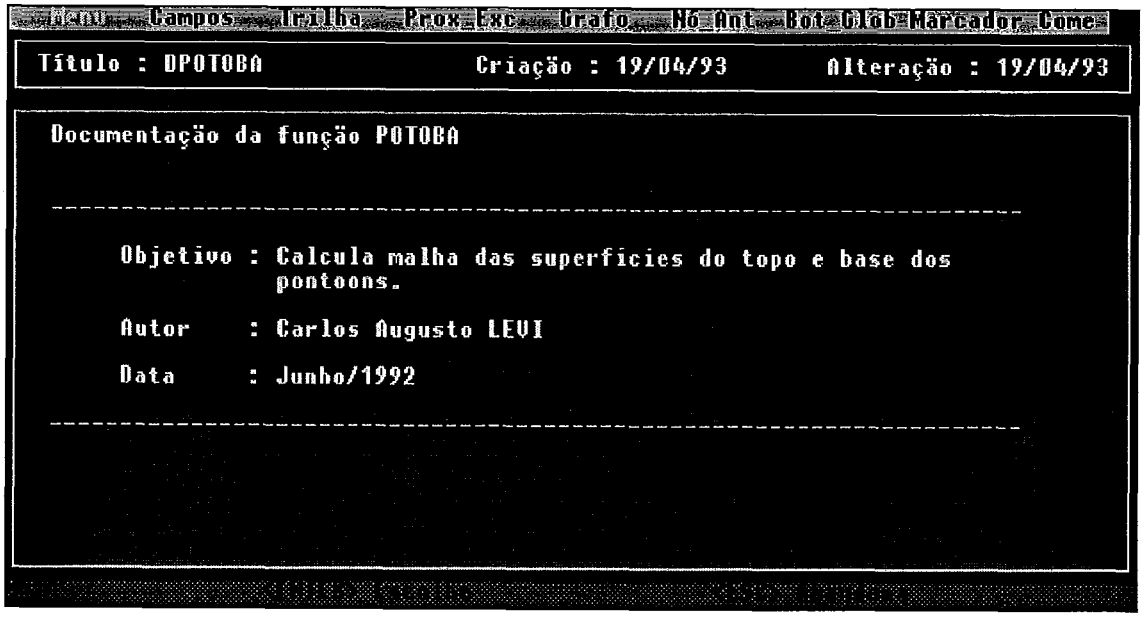


Figura 6.26. Navegação: Tela apresentada após retorno ao nó intermediário do módulo Potoba e seleção do botão *A documentação*

VI.2.1.4. Opção *Relatórios*

A execução desta opção equivale a apresentação de algumas subopções (ver figura 6.27).

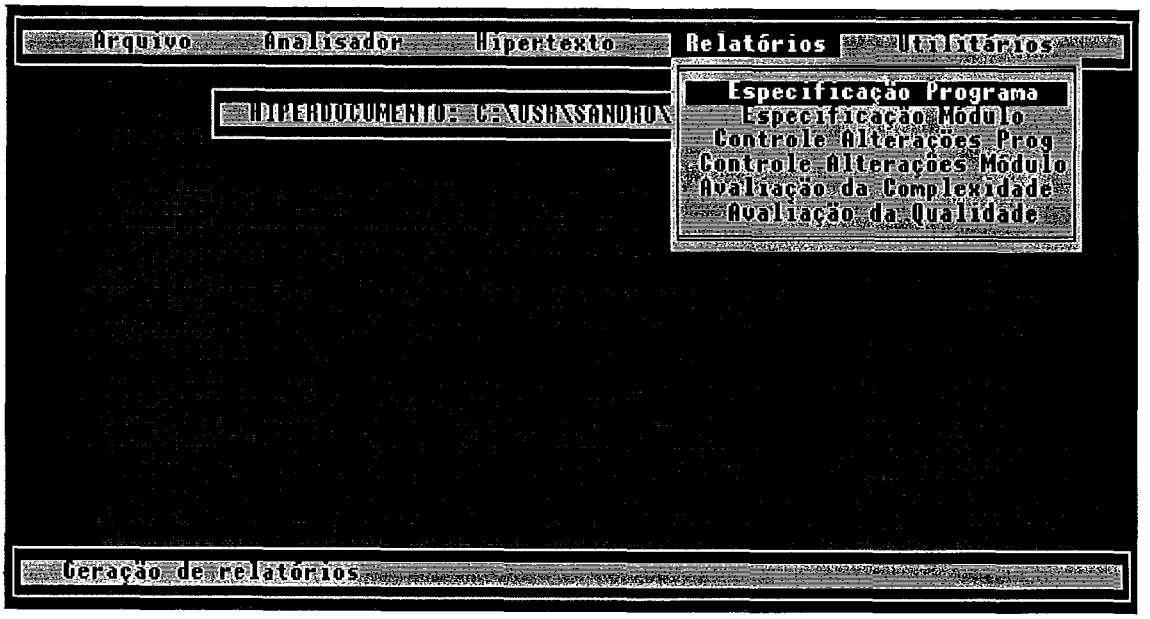


Figura 6.27. Opção *Relatórios*: Tela de subopções

VI.2.1.4.1. Subopção *Especificação Programa*

Esta subopção é selecionada quando o usuário deseja gerar o relatório de Especificação do Programa.

O apêndice D apresenta o relatório de Especificação do Programa para o programa MESH.

VI.2.1.4.2. Subopção *Especificação Módulo*

Esta subopção é selecionada quando o usuário deseja gerar o relatório de Especificação de um Módulo.

O apêndice D apresenta o relatório de Especificação de um Módulo do programa MESH (subrotina POTOBA).

VI.2.1.4.3. Subopção *Controle Alterações Prog*

Esta subopção é selecionada quando o usuário deseja gerar o relatório de Controle de Alterações do Programa.

O apêndice D apresenta o relatório de Controle de Alterações do Programa para o programa MESH, tendo sido simuladas algumas alterações nos módulos MESH e POTOBA.

VI.2.1.4.4. Subopção *Controle Alterações Módulo*

Esta subopção é selecionada quando o usuário deseja gerar o relatório de Controle de Alterações de um Módulo.

O apêndice D apresenta o relatório de Controle de Alterações de um módulo para o programa MESH, tendo sido simuladas algumas alterações no módulo POTOBA.

VI.2.1.4.5. Subopção *Avaliação da Complexidade*

Esta subopção é selecionada quando o usuário deseja imprimir o relatório de Avaliação da Complexidade gerado pelo ACF. Este relatório pode ter sido criado na subopção *Análise da Complexidade* ou durante a geração/regeração de um hiperdocumento.

VI.2.1.4.6. Subopção *Avaliação da Qualidade*

Esta subopção é selecionada quando o usuário deseja imprimir o relatório de Avaliação da Qualidade gerado pelo ACF. Este relatório pode ter sido criado na subopção *Análise da Qualidade* ou durante a geração/regeração de um hiperdocumento.

VI.2.1.5. Opção *Utilitários*

A execução desta opção equivale a apresentação de algumas subopções (ver figura 6.28).

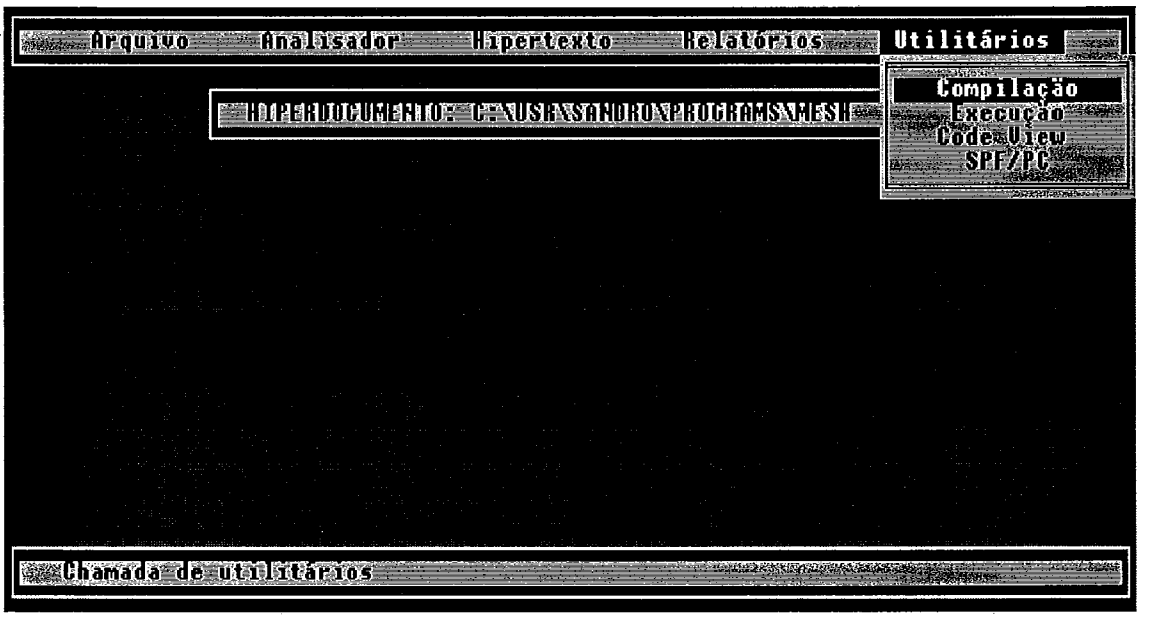


Figura 6.28. Opção *Utilitários*: Tela de subopções

VI.2.1.5.1. Subopção *Compilação*

Esta subopção é selecionada quando o usuário deseja compilar o programa .FOR que foi definido no módulo de definição do HyFor.

A execução desta subopção equivale a apresentação da tela mostrada na figura 6.29, onde o usuário deverá preencher os parâmetros de compilação do programa.

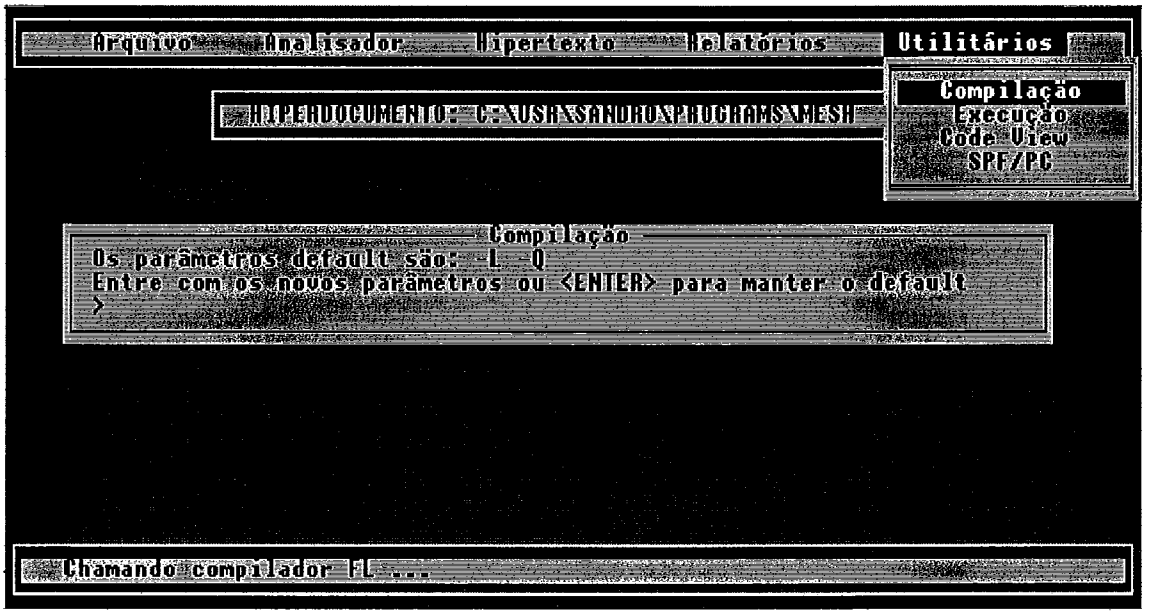


Figura 6.29. Subopção *Compilação*: Tela para obter os parâmetros de compilação

VI.2.1.5.2. Subopção *Execução*

Esta subopção é selecionada quando o usuário deseja executar o programa .EXE .FOR que foi definido no módulo de definição do HyFor.

VI.2.1.5.3. Subopção *Code View*

Esta subopção é selecionada quando o usuário deseja executar o utilitário CodeView para depuração do programa .FOR que foi definido no módulo de definição do HyFor.

VI.2.1.5.4. Subopção *SPF/PC*

A seleção desta subopção equivale a execução do SPF/PC. O HyFor acessa diretamente o módulo de edição deste utilitário, colocando como parâmetro de entrada para o editor, o diretório e os módulos .FOR que compõem o programa que foi definido no módulo de definição do HyFor.

O HyFor não impede e nem é prejudicado com a utilização deste utilitário sobre outros programas ou arquivos, dentro do ambiente.

A figura 6.30 apresenta a primeira tela mostrada do SPF/PC, quando este é acessado pelo HyFor. Esta tela apresenta a lista de todos os módulos que compõem o programa para que o usuário realize a seleção de edição.

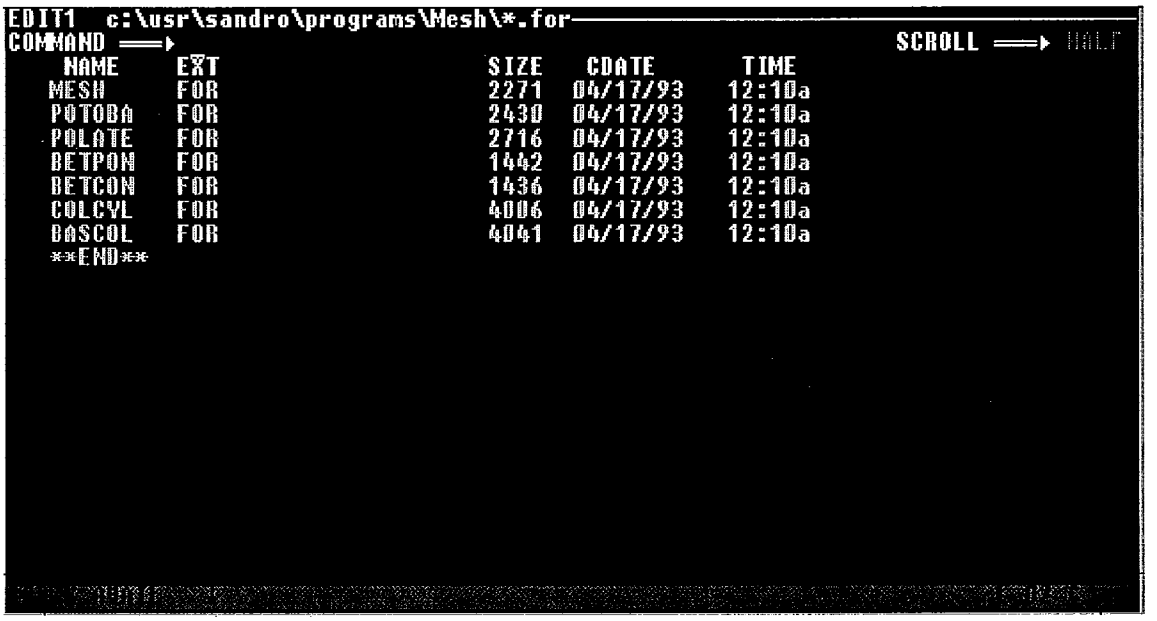


Figura 6.30. Subopção *SPF/PC*: Tela para seleção do módulo a ser editado

VII - Conclusões

O capítulo VII apresenta um resumo da proposta desta tese, relaciona os resultados obtidos com os objetivos a serem alcançados e identifica algumas propostas para trabalhos futuros.

VII.1. Conclusões

Levando-se em conta o alto custo do ciclo de vida do software científico devido aos problemas enfrentados durante seu desenvolvimento, foi proposto nesta tese um ambiente de apoio ao desenvolvimento e manutenção deste tipo de software. Este ambiente se propõe a aumentar efetivamente a produtividade do trabalho do programador de software científico, principalmente na fase de manutenção.

O ambiente está baseado na filosofia de hipertextos, promovendo a interligação de documentos do programa, permitindo a visualização das informações de forma conjunta e não linear. Hipertexto se mostrou um enfoque bastante adequado a esse objetivo.

Não foi estabelecido nenhum critério ou roteiro de documentação a ser seguido obrigatoriamente pelo especialista. Os documentos gerados contêm informações extraídas automaticamente do código ou informações acrescentadas pelo especialista segundo um formato sugerido, mas não obrigatório. Desta forma, o ambiente é plenamente utilizável para programas já existentes, sem que o especialista precise alterar sobremaneira seus hábitos de desenvolvimento.

Sobre este aspecto, é importante observar que não existe uma crítica favorável ou desfavorável a tais hábitos, apenas o ambiente proposto tenta se adaptar a esta rotina visando a sua efetiva utilização. Esta intenção se baseia no fato de já existirem propostas de métodos e ferramentas para suporte ao desenvolvimento de software em ambientes científicos e estas não terem sido amplamente reconhecidas por exigirem mudança na rotina e forma de trabalho do especialista.

Tendo em vista este aspecto, foi necessário considerar também a inclusão, no ambiente, das ferramentas de apoio a codificação em Fortran (compilador e depurador) e do editor de programas mais amplamente utilizado em áreas científicas (SPF/PC) para microcomputador.

O ambiente proposto levou em conta tanto a resistência a mudança do especialista quanto uma metodologia limitadamente informal.

A utilização do HyFor facilitará o entendimento do código e a geração automática de documentação atualizada, permitindo que o tempo gasto nas atividades de manutenção diminua. Desta forma, esta atividade poderá, inclusive, ser delegada a outros profissionais que não tenham participado do desenvolvimento e que possuam um custo/hora mais barato, deixando o especialista menos preso a tarefa de manutenção.

A facilidade de geração automática de relatórios que avaliam a qualidade do software e a possibilidade de visualização destes juntamente com o código, em um ambiente hipertexto, estimulará o especialista a analisá-los, fazendo com este possa vir a melhorar seus hábitos de desenvolvimento, aumentando a qualidade do produto final gerado, que repercutirá, diretamente, nos custos da fase de manutenção.

O protótipo desenvolvido para o HyFor atingiu os requisitos especificados para o ambiente e consideramos que sua utilização poderá realmente amenizar os problemas enfrentados no desenvolvimento e manutenção de software científico.

VII.2 Problemas enfrentados

Alguns problemas surgiram durante o desenvolvimento do HyFor e foram conseqüência de diversos fatores. O desenvolvimento em ambiente DOS para microcomputadores impôs vários limites de hardware que precisaram ser profundamente analisados.

O sistema hipertexto utilizado, o Hiperbase, não se mostrou eficiente no gerenciamento de um número muito grande de nós bem como para o controle de nós que possuam um tamanho acima de três páginas.

A utilização de módulos independentes para autoria e navegação do hiperdocumento torna o programa pouco ameno ao uso, podendo ser um motivo de desestímulo para o usuário.

VII.3. Sugestões para trabalhos futuros

A implementação de um sistema hipertexto específico para o ambiente contornará as dificuldades listadas acima e poderá considerar características específicas da área de aplicação.

O desenvolvimento de uma versão do HyFor para computador de grande porte poderá obter bastante sucesso, levando-se em conta a existência, neste ambiente, de um grande número de programas intensivamente utilizados em áreas científicas que enfrentam grandes dificuldades em sua manutenção.

Considerar esta proposta para outros tipos de ambientes de desenvolvimento, onde sejam utilizadas outras linguagens de programação, o COBOL, por exemplo, também é uma boa linha de estudo a ser seguida, tendo em vista que basicamente todos os ambientes enfrentam problemas no desenvolvimento e manutenção de software.

Referências Bibliográficas

[AKSC88] Akscyn, R., McCracken, D., Yoder, E., "KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations", Communications of the ACM, vol.31, num 7, pp.820-835, julho/1988.

[ANDR91] Andrade, C.J., "Avaliação da Qualidade de Programas", Dissertação de Tese de Mestrado, COPPE/UFRJ, 1991.

[ARAU89] Araujo Filho, H.A., definição informal transmitida no CENPES (Centro de Pesquisas da PETROBRAS), maio/1989.

[ARSA84] Arsac, J., "Informatique et Calcul Scientific" em "Tools, Methods and Languages for Scientific and Engineering Computation", (eds.) Ford, B. e outros, Elsevier Science Publishers, 1984.

[BAHI88] Bahia, A.S., "Controle de Qualidade de Software para a Área Científica", COPPE, ES-170/88, outobro/1988.

[BAHI92] 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.

[BEGE88] Begeman, M.L., Conklin, J., "The Right Tool for the Job", BYTE 13, pp.255-266, outubro/1988.

[BEIZ84] Beizer, B., Software System Testing and Quality Assurance, Van Nostrand Reinhold Company, 1984.

[BELL86] Bell, K., "Some Thoughts on Design, Development and maintenance of Engineering Software", Adv. Eng. Software, vol.8, 1986.

[BENN91] Bennett, K.H., "Automated support of software maintenance", Information and Software Technology, vol.33, num 1, janeiro/ fevereiro/1991.

[BERN84] Berns, G.M., "New Life for Fortran", Datamation, pp.166-174, setembro/1984.

[BIGE87] Bigelow, J. e Riley, V., "Manipulating Source Code in DynamicDesign", Hypertext'87 Papers,ACM, novembro/1987.

[BIGE88] Bigelow, J.,"Hypertext and CASE", IEEE Software, vol.5, num 2, pp.23-27, março/1988.

[BIGG89] Biggerstaff, T.J., Richter, C., "Reusability, Framework, Assessment, and Directions", em "Software Reusability", vol.I, (ed.) Biggerstaff, T.J., Perlis, A.J., Addison Wesley, 1989.

[BLAC87] Black, J.B., Carroll, J.M. e McGuigan,S.M., "What Kind of Minimal Instruction Manual is the Most Effective", ACM, 1987.

[BORG91] Borges, M.R.S., "Hipertextos: A próxima revolução no processo de desenvolvimento de sistemas de informação, XXIV Congresso Nacional de Informátic, 1991.

[BORG92] Borges, M.R.S., Notas do curso Tópicos Especiais em Engenharia de Software V - Hipertexto, COPPE/UFRJ, 1992.

[BROC76] Brockmann, R.J., "Writing Better Computer User Documentation: From Paper to Online", John Wiley & Sons, New York, 1986.

[BROW74] Brown, P.J., "Programming and Documenting Software Projects", Computing Surveys, Vol.6, Num 4, dezembro/1974.

[BROW87] Brown, P.J., "Turning ideas into prooducts: The Guide System", Proc. ACM Hypertext'87 Conf., pp.33-40, 1987

[BUEN89] Bueno, M.C.F., "Recomendações para Documentação de Programas e Sistemas", SUCESU, 1989.

[BUSH45] Bush, V., "As we may think", Atlantic Monthly, pp.101-108, julho/1945.

[CAMP88] Campbell, B., Goodman, J.M., "HAM: A general purpose hypertext abstract machine", Communications of the ACM vol.31, num 7, pp.856-861, julho/1988.

[CANN72] Canning, R., "The Maintenance 'Iceberg'", EDP Analyser, vol.10, num.10, outubro/1992.

[CARL87] Carle, A., "A practical Environment for Scientific Programming", Computer, janeiro/1987.

[CAVA92] Cavalcanti, M.C.R., "Manutenção: problemas e soluções", monografia do curso de Engenharia de Software, COPPE/UFRJ, 1992.

[CHEA89] Cheatham Jr., T.E., "Reusability through Program Transformations", em "Software Reusability", vol.I, (ed.) Biggerstaff,T.J., Perlis, A.J., Addison Wesley, 1989

[CONK87] Conklin, J., Begeman, M.L., "gIBIS: A hypertext tool for exploratory policy discussion", ACM Trans, Office Information systems, vol.6, num 4, pp.303-331, outubro/1988.

[CROS86] Cross, M., Moscardini, A.O. e Lewis, B. A., "Software Engineering Methodologies for Scientific and Engineering Computation", Appl. Math. Modelling, vol.10, outubro/1986.

[CUNT91] Cunto, W., Araujo, J., Giovannetti, F e Rivero, J., "FPLUS: Programming Environment for Scientific Applications", IEEE Software, setembro/1991.

[DEMA79] DeMarco, T., Structured Analysis and System Specification, Prentice-Hall, Inc., 1979.

[DEUT82] Deutsch, M.S., Software Verification and Validation, Prentice-Hall Inc., 1982.

[DIPO89] D'Ipolitto, C., "Hipertexto: Uma visão geral", COPPE/UFRJ, ES-197/89, 1989.

[ENGE63] Engelbart, D.C., "A Conceptual Framework for the Augmentation of Man's Intellect", Vistas in Information Handing, vol1, Soartam Books, Londres, 1963.

[ENGE68] Engelbart, D.C., English, W.K., "A Research Center for Augmenting Human Intellect, Proc. of the Fall Joint Computer Conference, pp.395-410, 1968.

[FAIR85] Fairley, R. E., Software Engineering Concepts, McGraw-Hill, 1985.

[FRIS88] Frisse, M.E., "Searching for Infoormation in a Hypertext Medical Handbook", Communications of the ACM, vol.31, num 7, pp.880-886, julho/1988.

[GALL91] Gallagher, D.B., Lyle, J.R., "Using Program Slicing in Software Maintenance", IEEE Transactions on Software Engineering, agosto/1991.

[GAMA89] Gama, C., "Manual de Práticas Recomendadas para Programação Fortran", versão preliminar, PETROBRAS/CENPES/DIPREX/ SEDEM, 1989.

[GARG89] Garg, P.K., Scacchi,W., "A Hypertext System to Manage Software Life Cycle Documents", IEEE Software, vol.5, 1989.

[GHEZ91] Ghezzi, C., Jazayeri, M., Mandrioli, D., Fundamentals of Software Engineering, Prentice Hall International, Inc., 1991.

[GORL91] Gorla, N., "Techniques for Application Software Maintenance", Information and Software Technology, vol.33, num 1, janeiro/ fevereiro/ 1991.

[GRIC88] Grice R.A.,"Information Development Is Part of Product Development - Not an Afterthought", Text, Context, and Hypertext: Writing with and for the computer, Edward Barret editor, MIT Press, Massachusetts,EUA, 1988.

[HALA88] Halasz, F.G., "Reflections on NoteCards: seven issues for the next generation of hypermedia systems", Communications of the ACM, vol.31, num 7, pp.836-852, julho/1988.

[HALA91] Halasz, F.G., "Tirania das Ligações", 1991.

[IEEE83] IEEE, "Standard glossary of software engineering terminology", ANSI/IEEE Standard 729 IEEE, 1983.

[ILES86] Iles,R.M.J.,Hague, S.J.,"Toolpack:The First Public Release", Computer Physics Communications 41, North-Holland, Amsterdam, 1986.

[JACK75] Jackson, M., Principles of Program Design, Academic Press, 1975.

[KEMM90] Kemmerer, R.A., "Integrating Formal Methods into the Development Process, IEEE Software, pp.37-50, setembro/1990.

[KRON87] Kronmeyer Filho, O.R., "Os Desafios em Documentação de Sistemas", ABACUS, setembro/1987.

[LIEN81] Lientz, B.P., Swanson, B.E., "Problems in Application Software Maintenance", Communications of ACM, vol 24(11), pp.763-769, novembro 1981.

[LIMA89] "Hipertexto e suas Aplicações", Projeto Final de Curso, Informática/IM/UFRJ, maio/1989.

[MAID88] Maidantchik, C.L.L., "Controle de Qualidade de Software Científico", COPPE/UFRJ, ES-167/88, setembro/1988.

[MAID92] Maidantchik, C.L.L., "SIM: Um Gerador Semi-Automático de Documentos", Dissertação de Tese de Mestrado, COPPE/UFRJ, março/1992.

[MORE88] Moreton,R., Second UK Software Maintenance Workshop, setembro/1988.-

[MULL88] Muller, H.A., "Rigi: A System for Programming_in_the_large", Proceedings of the 10 th Inter. cont. on Software Engineering, IEEE, pp.80-86, 1988.

[NIEL90] Nielsen, J., Hypertext and Hypermedia, Academic Press, Inc., 1990.

[OLIV85] Oliveira, J.R.L., Rocha, A.R.C., "O Ambiente de Programação e o software suscetível de mudança", XVIII Congresso Nacional de Informática, São Paulo, setembro/1985.

[PAGE88] Page-Jones, M., Projeto Estruturado de Sistemas, McGraw-Hill, 1988.

[PARI83] Parikh, G., Zvegintzov, N., Tutorial on Software Maintenance, Computer Society Press, 1983.

[PARN86] Parnas, D.L. e Clements, P.C., "A Rational Design Process: How and Why to Fake It", IEEE Transactions on Software Engineering, vol. SE-12, num 2, fevereiro/1986.

[PERE87] Pereira, J.C., "Metodologia para Desenvolvimento de Software para Engenharia", VIII Congresso Latino Americano e Ibérico, trab B-3, pp. 31-47, novembro/1987.

[PRES87] Pressman,R.S., SoftwareEngineering - A Practitioner's Approach, McGraw-Hill, 1987.

[PRIC89] Price, R.T., Golendziner, L.G., ""Banco de Dados para aplicações não convencionais", IV Escola Brasileira de Informática, janeiro/1989.

[RADA89] Rada, R., "Writing and Reading Hypertext: An Overview", Perspectives on Hypertext, ed. Lois F. Lunin, 1989.

[RICE71] Rice, J.R. (ed.), "Mathematical Software", Academic Press, New York, 1971.

[ROCH83] Rocha, A.R.C., "Um Modelo para Avaliação da Qualidade de Especificações", Dissertação de Tese de Doutorado, PUC-RJ, 1983.

[ROCH87A] Rocha, A.R.C., Aguiar, T.C., Nascimento, E.M. e Ferreira, R.L., "Algoritmos Numéricos X Software Numérico: Uma Visão do Engenheiro de Software", VIII Congresso Latino Americano e Ibérico, trab B-4, pp. 49-58, novembro/1987.

[ROCH87B] Rocha, A.R.C., Aguiar, T.C., Nascimento, E.M. e Ferreira, R.L., "Software Numérico: Contribuição da Engenharia de Software para a Qualidade do Produto".

[ROCH87C] Rocha, A.R.C., Análise e Projeto Estruturado de Sistemas, editora Campus, 1987.

[ROCH88A] Rocha, A.R.C., Relatório da 2a Fase do Projeto "Metodologia Para Desenvolvimento de Software Científico", COPPETEC ET-21045 -"ESTUDO E AVALIAÇÃO DA SITUAÇÃO ATUAL E DA AREA DE APLICAÇÃO", 1988.

[ROCH88B] Rocha, A.R.C., Relatório da 7a Fase do Projeto "Metodologia Para Desenvolvimento de Software Científico", COPPETEC ET-21045 - "RELATORIO FINAL", 1988.

[ROCH88C] Rocha, A.R.C. e Souza, J.M. (Editores) - "Ambientes de Desenvolvimento de Software e o Projeto TABA", COPPE, Seminário do Programa de Engenharia de Sistemas e Computação, resumos dos trabalhos apresentados, dezembro/1988.

[RUBE88] Rubens, P. e Krull, R., "Designing Online Information", Text, Context, and Hypertext: Writing with and for the computer, Edward Barret editor, MIT Press, Massachusetts, EUA, 1988.

[SCAC89] Scacchi, W., "The USC System Factory Project", Software Engineering Notes, vol14, num 1, pp.61-82, janeir/1989.

[SCHU91] Schutt, H.A., Streitz, N.A., "HyperBase: A Hypermedia Engine Based on a Relational Database Management System", Integrated Publication and Information Systems Institute, 1991.

[SHIR88] Shirk, H.N.,"Technical Writers as Computer Scientists: The Challenges of Online Documentation", Text, Context, and Hypertext: Writing with and for the computer, Edward Barret editor, MIT Press, Massachusetts,EUA, 1988.

[SHNE89] Shneiderman, B., "Reflections on authoring, editing, and managing hypertext", The society of text, Barrett, E. (ed.), MIT Press, Cambridge, MA, pp.115-131, 1989.

[SMIT90] Smith, D.B. e Oman, P.W., "Software Tools in Context", IEEE Software, vol.7, no.3, maio/1990.

[SWAN76] Swanson, E.B., "The Dimension of Maintenance", Proc. 2nd Int. Conf. Software Engineering IEEE, pp.492-497, 1976.

[TAYL59] Taylor, E.S., "An Interim Report on Engineering Design", Massachussetts Institute of Technology, Cambridge, MA, 1959.

[THOM87] Thompson, Bev, Thompson Bill, "Hyping Text: Hypertext and Knowledge Representation", AI Expert, pp.25-28, agosto/1987.

[TRIG86] Trigg, R.H., Weiser, M., "TEXTNET: A network based approach to text handling", ACM Trans. Office Inf. Syst., vol.4, num 1, pp.1-23, janeiro/1986.

[WERN88A] Werner, C.M.L., "O Desenvolvimento de Software Científico no SEDEM (pesquisa de opinião), Relatório Interno, PETROBRAS/ CENPES/DIPREX/SEDEM, janeiro/1988.

[WERN88B] Werner, C.M.L., "Proposta para Elaboração do Manual de Desenvolvimento de Software no SEDEM, Relatório interno, PETROBRAS/CENPES/DIPREX/SEDEM, 1988.

[WERN92] Werner, C.M.L., "Reutilização de Software no Desenvolvimento de Software Científico", dissertação de tese de doutorado, COPPE/UFRJ, março/1992.

[YOUR75] Yourdon, E., Techniques of Program Structure and Design, Prentice-Hall, Inc., 1975.

[YOUR85] Yourdon, E., Structured Walkthroughs, Prentice-Hall, Inc., 1985.

APÊNDICE A - Relação de métricas de complexidade

A.1. Apresentação

Este apêndice apresenta as diversas métricas de complexidade passíveis de serem selecionadas para análise e que foram utilizadas pelo Avaliador de Complexidade do ACF. [BAHI92]

Bahia agrupou estas métricas segundo dois critérios: de acordo com a característica de complexidade que desejam medir e de acordo com níveis crescentes de caracterização da complexidade.

A.2. Caracterização por Tipo de Complexidade

A.2.1 - Métricas de Tamanho

Número de linhas de declaração de dados
Número de linhas executáveis
Número de linhas de código
Número de linhas de comentários
Número de linhas de programa
Número de linhas de continuação
Número de linhas fora
Número de linhas em branco
Número de linhas de diretivas para o compilador
Número de linhas totais
Número de operadores distintos
Número de operadores totais
Número de operandos distintos
Número máximo de operadores no FORMAT
Número médio de operadores no FORMAT
Frequencia de uso dos operandos
Lista de operadores distintos
Lista de operandos distintos

A.2.2 - Métricas de Fluxo de Controle

Número de GOTO
Número total de conectores lógicos (and, or not)
Número total de IF's
Número máximo de encadeamento de IF's
Número médio de encadeamento de IF's
Número total de DO
Número total de DO WHILE

Número total de comandos CASE
Número médio de cláusulas por CASE
Número total de cláusulas CASE

A.2.3 - Métricas de Estrutura Modular

Número de rotinas chamadas (fan-out)
Número de rotinas chamantes (fan-in)
Lista das rotinas chamadas (fan-out)
Lista de rotinas chamantes (fan-in)
Número de ENTRY POINT distintos
Número de EXIT e RETURN
Número total de módulos por componente

A.2.4 - Métricas de Estrutura de Dados

Número de variáveis globais não usadas
Número de variáveis globais totais que chegam (common e parâmetros)
Número de variáveis globais usadas
Extensão máxima das variáveis
Lista de COMMON usados por minuto
Lista de módulos que usam determinado COMMON
Número de variáveis globais que chegam via COMMON
Número de variáveis globais que chegam via lista de parâmetros
Número de variáveis globais usadas apenas uma vez
Número de varáveis globais usadas sem declaração de tipo
Número de variáveis locais usadas apenas uma vez
Número de variáveis locais declaradas
Número de variáveis locais declaradas e não usadas
Número de variáveis locais usadas
Número de variáveis locais usadas e não declaradas
Lista de variáveis globais que chegam via COMMON
Lista de variáveis globais que chegam via lista de parâmetros
Lista de variáveis globais usadas apenas uma vez
Lista de variáveis globais não usadas
Lista de variáveis globais sem declaração de tipo
Lista de variáveis globais totais que chegam (COMMON e parâmetros)
Lista de variáveis globais usadas
Lista de variáveis locais usadas apenas uma vez
Lista de variáveis locais declaradas
Lista de variáveis locais declaradas e não usadas
Lista de variáveis locais usadas
Lista de variáveis locais usadas e não declaradas

A.2.5 - Métricas de Entrada/Saída de dados

Número de variáveis de I/O
Número de arquivos acessados
Número de comandos FORMAT
Número de variáveis de I/O não utilizadas

Número de arquivos acessados para leitura
Número de arquivos acessados para atualização
Número de arquivos acessados para gravação
Lista de variáveis de I/O
Lista de variáveis de I/O não utilizadas
Lista de arquivos acessados para leitura
Lista de arquivos acessados para atualização
Lista de arquivos acessados para gravação

A.2.6 - Métricas de documentação

Percentual de linhas de comentário
Percentual linhas em branco

A.2.7 - Métricas agregadas

McCabe (IF's)
McCabe Modificado (IF's e conectores lógicos)
Henry e Kafura (var iáveis de entrada e saída, fan-in e fan-out)
Card e Agresti (var iáveis de entrada e saída, fan-in e fan-out)
Halstead (operandos e operadores)
Harrison e Cook (variáveis globais usadas, tamanho, McCabe)

A.3. Caracterização por Nível de Complexidade

A.3.1 - Grupo primário

McCabe (IF's)
McCabe Modificado (IF's e conectores lógicos)
Número de variáveis globais não usadas
Número de variáveis globais que chegam (COMMON e parâmetros)
Número de variáveis globais usadas
Número de GOTO
Número de conectores lógicos
Número total de IF's
Número de rotinas chamadas (fan-out)
Número de rotinas chamantes (fan-in)
Percentual de linhas de comentário
Percentual de linhas em branco
Número de variáveis de I/O
Número de arquivos acessados
Número de linhas de código (exec + decl)
Número de linhas de comentários (come)
Número de linhas de declaração de dados (decl)
Número de linhas de diretivas para o compilador (comp)
Número de linhas em branco (bran)
Número de linhas entregues (exec + decl + come)
Número de linhas executáveis (exec)

Numero de linhas (exec + decl + come + comp + bran)

A.3.2 - Grupo Mediano

inclui todos do grupo anterior e mais:

Card e Agresti (variáveis de entrada e saída, fan-out)
Halstead (operandos e operadores)
Harrison e Cook (variáveis globais usadas, tamanho, McCabe)
Henry e Kafura (variáveis de entrada e saída, fan-in, fan-out)
Extensão máxima das variáveis
Extensão média das variáveis
Lista de COMMON usados por módulo
Lista de módulos que usam determinado COMMON
Número de variáveis globais que chegam via COMMON
Número de variáveis globais que chegam via lista de parâmetros
Número de variáveis globais usadas apenas uma vez
Número de variáveis globais sem declaração de tipo
Número de variáveis locais usadas apenas uma vez
Número de variáveis locais declaradas
Número de variáveis locais declaradas e não usadas
Número de variáveis locais usadas
Número de variáveis locais usadas e não declaradas
Número máximo de encadeamento de IF's
Número médio de encadeamento de IF's
Número total de DO's
Número total de DO WHILE
Número total de comandos CASE
Lista de rotinas chamadas (fan-out)
Lista de rotinas chamantes (fan-in)
Número de ENTRY POINT distintos
Número de EXIT e RETURN's
Número total de módulo por componente
Número de comandos FORMAT
Número de variáveis de I/O não utilizadas
Número de arquivos acessados para leitura
Número de arquivos acessados para atualização
Número de arquivos acessados para gravação
Número de operadores distintos
Número de operadores totais
Número de operandos distintos
Número de operandos totais
Número máximo de operadores no format
Número médio de operadores no format

A.3.3 -Grupo Extenso

inclui todas do grupo anterior e mais:

Lista de variáveis globais que chegam via COMMON
Lista de variáveis globais que chegam via lista de parâmetros
Lista de variáveis globais usadas apenas uma vez
Lista de variáveis globais não usadas

Lista de variáveis globais sem declaração de tipo
Lista de variáveis globais totais que chegam (COMMON e parâmetros)
Lista de variáveis globais usadas
Lista de variáveis locais usadas apenas uma vez
Lista de variáveis locais declaradas
Lista de variáveis locais declaradas e não usadas
Lista de variáveis locais usadas
Lista de variáveis locais usadas e não declaradas
Número médio de cláusulas por CASE
Número total de cláusulas CASE
Lista de variáveis de I/O
Lista de variáveis de I/O não utilizadas
Lista de arquivos acessados para leitura
Lista de arquivos acessados para atualização
Lista de arquivos acessados para gravação
Lista de arquivos acessados
Frequencia de uso dos operandos
Lista de operadores distintos
Lista de operadores
Lista de operandos didstintos
Lista de operandos

APÊNDICE B - Relatório de Avaliação da Complexidade

B.1. Apresentação

Este apêndice apresenta um exemplo do Relatório de Avaliação da Complexidade gerado pelo ACF. Este foi gerado a partir da análise do programa MESH.

B.2. Relatório

Relatorio de Avaliacao da Complexidade

Data: 19/04/93
Usuario: Carla

Modulo	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
Media	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. nao usadas
C - Vars. Gbls. que chegam
E - Vars. Gbls. usadas
F - Numero de Goto's
G - Numero de Con. Logicos
H - Percentual de Comentarios
I - Percentual de Brancos
J - Numero de Vars. de E/S
L - Linhas de Codigo
M - Linhas de Comentario
N - Linhas de Decl. de Dados
O - Diretivas ao Compilador
P - Linhas em Branco
Q - Linhas Entregues
R - Linhas Executaveis
S - Total de Linhas
T - Operandos Distintos
U - Total de Operandos
V - Operadores Distintos
X - Total de Operadores

APÊNDICE C - Relatório de Avaliação da Qualidade

C.1. Apresentação

Este apêndice apresenta um exemplo do Relatório de Avaliação da Qualidade gerado pelo ACF. Este foi gerado a partir da análise do programa MESH.

C.2. Relatório

Relatorio de Avaliacao da Qualidade

Data: 19/04/93
Usuario: carla

Fase 1: Verificacao das Descricoes dos Modulos.
Nesta fase sao verificados os comentario de descricao de cada modulo ou possiveis candidatos fora de posicao (entre modulos)

Fase 2: Verificacao da Legibilidade do Programa
Nesta fase sao analisadas as falhas de identacao bem como praticas pouco recomendadas de programacao tais como o uso de DIMENSION para declarar vetores e matrizes

- O modulo MESH faz uso de DIMENSION, o que eh pouco recomendado
- O DO da linha 27 nao eh terminado por um CONTINUE
- O bloco da linha 28 tem indentacao insuficiente
- O DO da linha 29 nao eh terminado por um CONTINUE
- O bloco da linha 30 tem indentacao insuficiente
- O modulo POTOBA faz uso de DIMENSION, o que eh pouco recomendado
- O bloco da linha 41 tem indentacao insuficiente
- O DO da linha 47 termina no mesmo ponto que o da linha 40
- O bloco da linha 48 tem indentacao insuficiente
- O bloco da linha 59 tem indentacao insuficiente
- O DO da linha 61 termina no mesmo ponto que o da linha 58
- O bloco da linha 63 tem indentacao insuficiente
- O bloco da linha 73 tem indentacao insuficiente
- O DO da linha 72 nao eh terminado por um CONTINUE
- O bloco da linha 80 tem indentacao insuficiente
- O DO da linha 79 nao eh terminado por um CONTINUE
- O bloco da linha 92 tem indentacao insuficiente
- O DO da linha 91 nao eh terminado por um CONTINUE
- O bloco da linha 99 tem indentacao insuficiente
- O DO da linha 98 nao eh terminado por um CONTINUE
- O modulo POLATE faz uso de DIMENSION, o que eh pouco recomendado
- O bloco da linha 124 tem indentacao insuficiente
- O DO da linha 127 termina no mesmo ponto que o da linha 123

- O bloco da linha 128 tem indentacao insuficiente
- O bloco da linha 139 tem indentacao insuficiente
- O DO da linha 141 termina no mesmo ponto que o da linha 138
- O bloco da linha 143 tem indentacao insuficiente
- O bloco da linha 156 tem indentacao insuficiente
- O DO da linha 155 nao eh terminado por um CONTINUE
- O bloco da linha 163 tem indentacao insuficiente
- O DO da linha 162 nao eh terminado por um CONTINUE
- O bloco da linha 175 tem indentacao insuficiente
- O DO da linha 174 nao eh terminado por um CONTINUE
- O bloco da linha 183 tem indentacao insuficiente
- O DO da linha 182 nao eh terminado por um CONTINUE
- O modulo BETPON faz uso de DIMENSION, o que eh pouco recomendado
- O bloco da linha 206 tem indentacao insuficiente
- O DO da linha 209 termina no mesmo ponto que o da linha 205
- O bloco da linha 210 tem indentacao insuficiente
- O DO da linha 223 termina no mesmo ponto que o da linha 220
- O modulo BETCON faz uso de DIMENSION, o que eh pouco recomendado
- O bloco da linha 246 tem indentacao insuficiente
- O DO da linha 249 termina no mesmo ponto que o da linha 245
- O bloco da linha 250 tem indentacao insuficiente
- O DO da linha 263 termina no mesmo ponto que o da linha 260
- O modulo COLCYL faz uso de DIMENSION, o que eh pouco recomendado
- O DO da linha 295 termina no mesmo ponto que o da linha 290
- O DO da linha 308 termina no mesmo ponto que o da linha 305
- O comando da linha 329 nao retornou a indentacao original
- O DO da linha 335 termina no mesmo ponto que o da linha 332
- O DO da linha 350 termina no mesmo ponto que o da linha 345
- O DO da linha 363 termina no mesmo ponto que o da linha 360
- O DO da linha 376 termina no mesmo ponto que o da linha 373
- O DO da linha 389 termina no mesmo ponto que o da linha 386
- O modulo BASCOL faz uso de DIMENSION, o que eh pouco recomendado
- O DO da linha 419 termina no mesmo ponto que o da linha 412
- O DO da linha 432 termina no mesmo ponto que o da linha 429
- O DO da linha 445 termina no mesmo ponto que o da linha 442
- O DO da linha 458 termina no mesmo ponto que o da linha 455
- O DO da linha 475 termina no mesmo ponto que o da linha 468
- O DO da linha 488 termina no mesmo ponto que o da linha 485
- O DO da linha 501 termina no mesmo ponto que o da linha 498
- O DO da linha 514 termina no mesmo ponto que o da linha 511

Fase 3: Verificacao das Estruturas de Dados

Nesta fase sao listados os identificadores nao declarados e os blocos COMMON sao checados e suas redefinicoes sao apontadas

- Os identificadores abaixo nao foram declarados no modulo MESH:

A	B	I	R
T	NE	PI	NO
XL	NBH	NPA	NPC
NCR	NFR	NPL	NBHL

- Os identificadores abaixo nao foram declarados no modulo POTOBA:

A	B	I	J
R	T	NE	PI
RI	NO	XL	XR
NPC	NEO	COS	NPL
NOO	NPC1	NPL1	

- Os identificadores abaixo nao foram declarados no modulo POLATE:

A	B	I	J
R	T	NE	PI
RI	NO	XL	XR
KNE	NPA	NPC	COS
NOO	KNE2	NPA1	NPC1
NOAN1			

- Os identificadores abaixo nao foram declarados no modulo BETPON:

A	B	I	J
R	T	NE	DT
PI	RI	NO	XL
NBH	NPA	COS	NBH1
NPA1	ALFA		

- Os identificadores abaixo nao foram declarados no modulo BETCON:

A	B	I	J
R	T	NE	DT
PI	RI	NO	XL
NPA	COS	NPA1	ALFA
NBHL	NBHL1		

- Os identificadores abaixo nao foram declarados no modulo COLCYL:

A	B	I	J
R	T	NE	DT
PI	RI	NO	XL
XR	NBH	NCR	COS
NPL	NBH1	ALFA	NCR1
NPL1	NBHL	NBHL1	

- Os identificadores abaixo nao foram declarados no modulo BASCOL:

B	I	J	R
T	NE	DT	PI
RI	NO	XL	XR
NBH	COS	NFR	NPL
NBH1	ALFA	NFR1	NPL1
NBHL	NBHL1		

APÊNDICE D - Relatórios gerados pelo GAD

D.1. Apresentação

Este apêndice se destina a apresentar os relatórios gerados automaticamente pelo GAD, a partir do hiperdocumento.

Os relatórios apresentados, Especificação do Programa e Relatório de Controle de Alterações do Programa, por serem muito extensos, estão sendo parcialmente apresentados, sem afetar o objetivo da apresentação.

Os outros relatórios gerados pelo GAD, Especificação de um Módulo e Relatório de Controle de Alterações de um Módulo, não são apresentados por serem subconjuntos dos relatórios citados acima, respectivamente.

D.2. Especificação do Programa

```
==========================================================================

                HyFor - Módulo de Geração Automática de Documentação
                ESPECIFICAÇÃO DO PROGRAMA MESH

==========================================================================

1) Identificação do Programa
   ---------------------------------
        Nome do Programa: MESH
        Autor do Hiperdocumento: CARLA
        Data de criação do hiperdocumento: 04/19/1993 às 22:47:10


2) Grafo de Chamada dos Módulos
   ----------------------------------------

        Módulo MESH
                Faz referência ...:
                        POTOBA
                        POLATE
                        BETPON
                        BETCON
                        COLCYL
                        BASCOL
                É Referenciado por:
```

Módulo POTOBA
Faz referência ...:
É Referenciado por:
MESH

Módulo POLATE
Faz referência ...:
É Referenciado por:
MESH

Módulo BETPON
Faz referência ...:
É Referenciado por:
MESH

Módulo BETCON
Faz referência ...:
É Referenciado por:
MESH

Módulo COLCYL
Faz referência ...:
É Referenciado por:
MESH

Módulo BASCOL
Faz referência ...:
É Referenciado por:
MESH

3) Descrição de Cada Módulo

Nome do Módulo: MESH
Descrição Geral:

Objetivo : Geracao de malha de plataformas tipo TLP ou semisubmersivel com colunas circulares e pontoons retangulares.

Algoritmo: Dadas as dimensoes principais a malha e calculada para as seguintes superficies:

1. Topo e base do pontoon
2. Laterais dos pontoons
3. Superficie da coluna entre pontoons(menor perimetro)
4. Superficie da coluna entre pontoons(maior perimetro)
5. Superficie da coluna acima dos pontoons
6. Superficie da coluna restante
7. Base da coluna

Imprime resultado no formato do programa "Genesis".

Autor : Luiz Augusto LEVY

Data : junho/1992.

Dicionário de Dados:

Variável X da função MESH

Tipo: REAL
Declarada: SIM
Escopo: GLOBAL
Forma Global: PARAM
Operacoes de E/S: ESCRITA
Ocorrencias: 8
Linha da primeira referencia: 2
Linha da ultima referencia: 29
De onde vem:

Modulo	Sinomimo

Para onde vai:

Modulo	Sinomimo
POTOBA	X
POLATE	X
BETPON	X
BETCON	X
COLCYL	X
BASCOL	X

Variável R da função MESH

Tipo: REAL
Declarada: NAO
Escopo: GLOBAL
Forma Global: PARAM
Operacoes de E/S: LEITURA
Ocorrencias: 7
Linha da primeira referencia: 7
Linha da ultima referencia: 25
De onde vem:

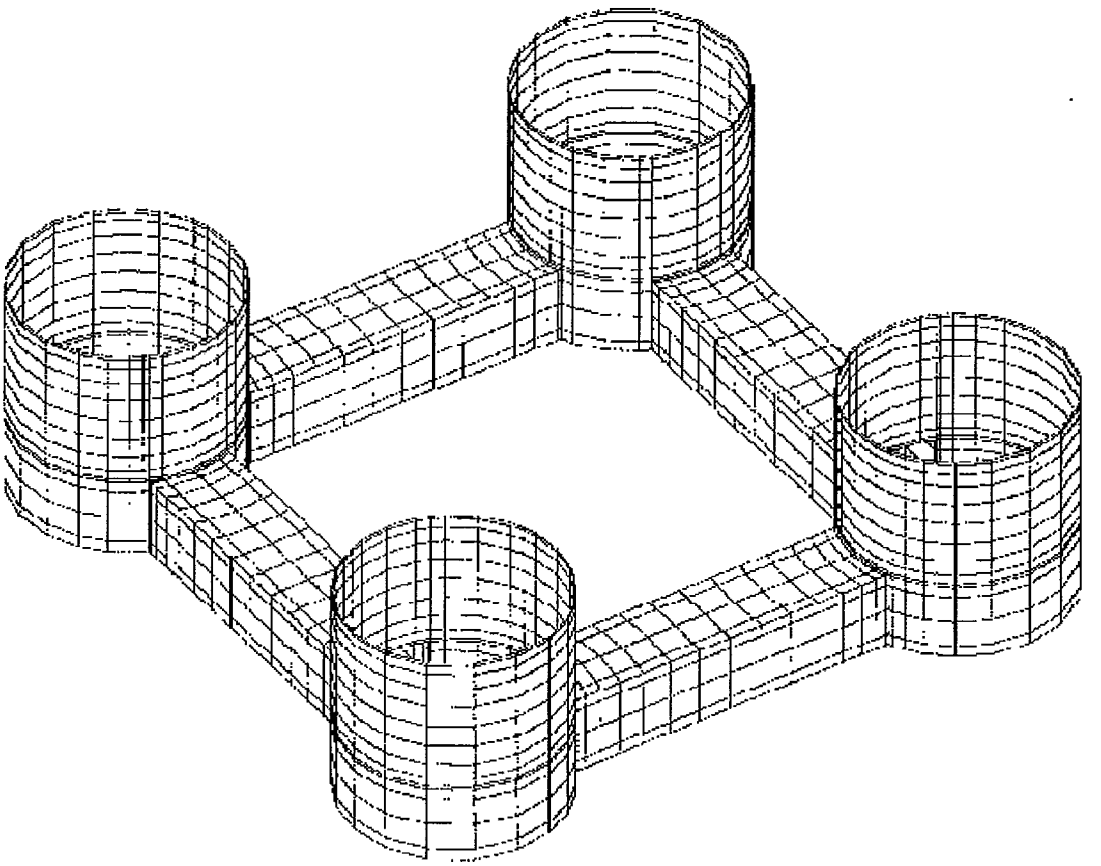
Modulo	Sinomimo

Para onde vai:

Modulo	Sinomimo
POTOBA	R
POLATE	R
BETPON	R
BETCON	R
COLCYL	R
BASCOL	R

Observações sobre a implementação:

Esquematização da estrutura da plataforma considerada pelo MESH:



Nome do Módulo: POTOBA
Descrição Geral:

Objetivo : Calcula malha das superficies do topo e base dos pontoons.

Autor : Carlos Augusto LEVI

Data : Junho/1992

Dicionário de Dados:

Variável PI da função POTOBA

Tipo: REAL
Declarada: NAO
Escopo: GLOBAL
Forma Global: PARAM
Operacoes de E/S:
Ocorrencias: 3
Linha da primeira referencia: 2
Linha da ultima referencia: 19
De onde vem:

Modulo	Sinomimo
MESH	PI

Para onde vai:

Modulo	Sinomimo

Variável B da função POTOBA

Tipo: REAL
Declarada: NAO
Escopo: GLOBAL
Forma Global: PARAM
Operacoes de E/S:
Ocorrencias: 3
Linha da primeira referencia: 2
Linha da ultima referencia: 13
De onde vem:

Modulo	Sinomimo
MESH	B

Para onde vai:

Modulo	Sinomimo

Observações sobre a implementação:

D.3. Relatório de Alterações do Programa

HyFor - Módulo de Geração Automática de Documentação
RELATÓRIO DE ALTERAÇÕES DO PROGRAMA MESH

1) Identificação do Programa

Nome do Programa: MESH
Autor do Hiperdocumento: CARLA
Data de criação do hiperdocumento: 04/19/1993 às 22:47:10

2) Descrição de cada módulo

Nome do Módulo: MESH
Descrição Geral:

Objetivo : Geracao de malha de plataformas tipo TLP ou semisubmersivel com colunas circulares e pontoons retangulares.

Algoritmo: Dadas as dimensoes principais a malha e calculada para as seguintes superficies:

1. Topo e base do pontoon
2. Laterais dos pontoons
3. Superficie da coluna entre pontoons(menor perimetro)
4. Superficie da coluna entre pontoons(maior perimetro)
5. Superficie da coluna acima dos pontoons
6. Superficie da coluna restante
7. Base da coluna

Imprime resultado no formato do programa "Genesis".

Autor : Luiz Augusto LEVY

Data : junho/1992.

Não foram feitas alterações neste módulo !

Nome do Módulo: POTOBA
Descrição Geral:

Objetivo : Calcula malha das superficies do topo e base dos pontoons.

Autor : Carlos Augusto LEVI

Data : Junho/1992

Alterações em relação a versão anterior:

Inclusão da variável AUX

Autor da documentação das alterações: CARLA
Data da documentação das alterações: 19/04/93

APÊNDICE E - Listagem do programa MESH

E.1. Apresentação

Este apêndice apresenta a listagem original do programa MESH, que foi usado no exemplo de utilização do HyFor.

E.2. Listagem

```
      PROGRAM MESH
C -----------------------------------------------------------------
C
C   Objetivo : Geracao de malha de plataformas tipo TLP ou
C              semisubmersivel com colunas circulares e pontoons
C              retangulares.
C
C   Algoritmo: Dadas as dimensoes principais a malha e calculada
C              para as seguintes superficies:
C
C              1. Topo e base do pontoon
C              2. Laterais dos pontoons
C              3. Superficie da coluna entre pontoons(menor perimetro)
C              4. Superficie da coluna entre pontoons(maior perimetro)
C              5. Superficie da coluna acima dos pontoons
C              6. Superficie da coluna restante
C              7. Base da coluna
C
C              Imprime resultado no formato do programa "Genesis".
C
C   Autor    : Luiz Augusto LEVY
C
C   Data     : junho/1992.
C
C -----------------------------------------------------------------
C
      DIMENSION X(4000),Y(4000),Z(4000),ICON(4000,4)
C
      OPEN (5,FILE='DADOS',FORM='FORMATTED',STATUS='OLD')
      OPEN (6,FILE='RESUL',FORM='FORMATTED',STATUS='UNKNOWN')
C
      READ (5,*) T,R,XL,A,B
      READ (5,*) NPL,NPC,NPA,NFR,NBH,NBHL,NCR
C
C     DEFINE VARIAVEIS
C
      PI = ACOS (-1.)
      NE = 0
      NO = 0
C
C     MALHA DAS SUPERFICIES DO TOPO E BASE DOS PONTOONS
C
      CALL POTOBA (PI,XL,A,B,R,T,NPL,NPC,NO,NE,X,Y,Z,ICON)
C
C     MALHA DAS SUPERFICIES LATERAIS DOS PONTOONS
C
      CALL POLATE (PI,XL,A,B,R,T,NPA,NPC,NO,NE,X,Y,Z,ICON)
C
C     MALHA DA SUPERFICIE DA COLUNA ENTRE PONTOONS
```

```
C
      CALL BETPON (PI,XL,A,B,R,T,NPA,NBH,NO,NE,X,Y,Z,ICON)
C
C     MALHA DA SUPERFICIE DA COLUNA COMPLEMENTAR A ENTRE PONTOONS
C
      CALL BETCON (PI,XL,A,B,R,T,NPA,NBHL,NO,NE,X,Y,Z,ICON)
C
C     MALHA DA SUPERFICIE DA COLUNA ACIMA DO PONTOON
C
      CALL COLCYL(PI,XL,A,B,R,T,NPL,NBHL,NBH,NCR,NO,NE,X,Y,Z,ICON)
C
C     MALHA DA SUPERFICIE DA BASE DA COLUNA
C
      CALL BASCOL(PI,XL,B,R,T,NFR,NPL,NBHL,NBH,NO,NE,X,Y,Z,ICON)
C
      WRITE (6,*) NO,NE
      DO 10 I = 1 , NO
   10 WRITE (6,*) I,X(I),Y(I),Z(I)
      DO 20 I = 1,NE
   20 WRITE (6,100) I,ICON(I,1),ICON(I,2),ICON(I,3),ICON(I,4)
  100 FORMAT (I5,'  2  ',4I5,'   0 0 0 0')
      STOP
      END
      SUBROUTINE POTOBA (PI,XL,A,B,R,T,NPL,NPC,NO,NE,X,Y,Z,ICON)
C -----------------------------------------------------------------
C
C     Objetivo : Calcula malha das superficies do topo e base dos
C                pontoons.
C
C     Autor    : Carlos Augusto LEVI
C
C     Data     : Junho/1992
C
C -----------------------------------------------------------------
C
      DIMENSION X(4000),Y(4000),Z(4000),ICON(4000,4)
C
C     COORDENADAS DOS PONTOS AO LONGO DA LARGURA
C
      NPL1 = NPL + 1
      NPC1 = NPC + 1
C
      DO 10 I = 1 , NPL1
      RI = FLOAT (I-1)
      XX = B * ABS (COS(RI*PI/NPL) -1.0) / 2.0
      XR = -B/2.0 + XX
      YR = SQRT (R*R - XR*XR)
      COMPI = XL - YR
C
C     COORDENADAS DOS PONTOS AO LONGO DO COMPRIMENTO
C
      DO 10 J = 1 , NPC1
      RJ = FLOAT (J-1)
      YY = COMPI * ABS (COS(RJ*PI/NPC/2.0) -1.0)
C
C     COORDENADAS DOS NOS DA BASE NO REFERENCIAL GLOBAL
C
      NO = NO + 1
      X(NO) = XL - XR
      Y(NO) = XL - YR - YY
      Z(NO) = - T
C
   10 CONTINUE
C
C     CONECTIVIDADE
C
      DO 20 I = 1 , NPL
      IR1 = (I-1) * NPC1
      IR2 =  I  * NPC1
      DO 20 J = 1 , NPC
C
```

```
      NE = NE + 1
      ICON (NE,1) = IR1 + J
      ICON (NE,2) = IR1 + J + 1
      ICON (NE,3) = IR2 + J + 1
      ICON (NE,4) = IR2 + J
C
 20   CONTINUE
C
C     COORDENADAS DOS NOS DO TOPO DO MESMO PONTOON
C
      NOO = NO
      DO 30 I = 1 , NOO
      NO = NO + 1
      X(NO) = X(I)
      Y(NO) = Y(I)
 30   Z(NO) = -T + A
C
C     CONECTIVIDADE DOS ELEMENTOS DO TOPO
C
      NEO = NE
      DO 40 I = 1 , NEO
      NE = NE + 1
      I1 = ICON (I,1)
      I2 = ICON (I,2)
      I3 = ICON (I,3)
      I4 = ICON (I,4)
      ICON (NE,1) = I1 + NOO
      ICON (NE,2) = I4 + NOO
      ICON (NE,3) = I3 + NOO
 40   ICON (NE,4) = I2 + NOO
C
C     COORDENADAS DOS NOS DA BASE E DO TOPO DO SEGUNDO PONTOON
C          ( ROTACAO -90 DEG. EM (XL,XL) )
C
      NOO = NO
      DO 50 I = 1 , NOO
      NO = NO + 1
      X (NO) =  Y (I)
      Y (NO) = - X (I) + 2.0 * XL
 50   Z (NO) =  Z (I)
C
C     CONECTIVIDADE DOS ELEMENTOS DO SEGUNDO PONTOON
C
      NEO = NE
      DO 60 I = 1 , NEO
      NE = NE + 1
      I1 = ICON (I,1)
      I2 = ICON (I,2)
      I3 = ICON (I,3)
      I4 = ICON (I,4)
      ICON (NE,1) = I1 + NOO
      ICON (NE,2) = I2 + NOO
      ICON (NE,3) = I3 + NOO
 60   ICON (NE,4) = I4 + NOO
C
      RETURN
      END
      SUBROUTINE POLATE (PI,XL,A,B,R,T,NPA,NPC,NO,NE,X,Y,Z,ICON)
C -------------------------------------------------------------------
C
C     Objetivo : Calculo da malha das superficies laterais dos pontoons
C
C     Autor    : Carlos Augusto LEVI
C
C     Data     : Junho/1992
C
C -------------------------------------------------------------------
C
      DIMENSION X(4000),Y(4000),Z(4000),ICON(4000,4)
C
C     COMPRIMENTO DA SUPERFICIE
```

```
C
      XR = -B/2.0 + XX
      YR = SQRT (R*R - XR*XR)
      COMPI = XL - YR
C
C     COORDENADAS DOS PONTOS AO LONGO DA LARGURA
C
      NPA1 = NPA + 1
      NPC1 = NPC + 1
      NOAN = NO
      NEAN = NE
C
      DO 10 I = 1 , NPA1
      RI = FLOAT (I-1)
      HH = A * ABS (COS(RI*PI/NPA) -1.0) / 2.0
C
C     COORDENADAS DOS PONTOS AO LONGO DO COMPRIMENTO
C
      DO 10 J = 1 , NPC1
      RJ = FLOAT (J-1)
      YY = COMPI * ABS (COS(RJ*PI/NPC/2.0) -1.0)
C
C     COORDENADAS DOS NOS DA LATERAL "+X"  NO REFERENCIAL GLOBAL
C
      NO = NO + 1
      X(NO) = XL - XR
      Y(NO) = XL - YR - YY
      Z(NO) = - T + HH
C
   10 CONTINUE
C
C     CONECTIVIDADE DOS ELEMENTOS DA LATERAL "+X"
C
      DO 20 I = 1 , NPA
      IR1 = (I-1) * NPC1 + NOAN
      IR2 =  I  * NPC1 + NOAN
      DO 20 J = 1 , NPC
C
      NE = NE + 1
      ICON (NE,1) = IR1 + J
      ICON (NE,2) = IR2 + J
      ICON (NE,3) = IR2 + J + 1
      ICON (NE,4) = IR1 + J + 1
C
   20 CONTINUE
C
C     COORDENADAS DOS NOS DA LATERAL "-X"  DO MESMO PONTOON
C
      KNE = NE - NEAN
      KNO = NO - NOAN
      NOAN1 = NOAN + 1
      NOO = NO
      DO 30 I = NOAN1 , NOO
      NO = NO + 1
      X(NO) = X(I) - B
      Y(NO) = Y(I)
   30 Z(NO) = Z(I)
C
C     CONECTIVIDADE DOS ELEMENTOS DA LATERAL "-X"
C
      NEO = NE
      DO 40 I = 1 , KNE
      NE = NE + 1
      I1 = ICON (I+NEAN,1)
      I2 = ICON (I+NEAN,2)
      I3 = ICON (I+NEAN,3)
      I4 = ICON (I+NEAN,4)
      ICON (NE,1) = I1 + KNO
      ICON (NE,2) = I4 + KNO
      ICON (NE,3) = I3 + KNO
   40 ICON (NE,4) = I2 + KNO
```

```
C
C   COORDENADAS DOS NOS DA LATERAL DO PONTOON  "+/-Y"
C         ( ROTACAO -90 DEG. EM (XL,XL) )
C
    NOO = NO
    DO 50 I = NOAN1 , NOO
    NO = NO + 1
    X (NO) =   Y (I)
    Y (NO) = - X (I) + 2.0 * XL
 50 Z (NO) =   Z (I)
C
C   CONECTIVIDADE DOS ELEMENTOS DA LATERAL DO SEGUNDO PONTOON
C
    KNE2 = 2.*KNE
    KNO2 = 2.*KNO
    DO 60 I = 1 , KNE2
    NE = NE + 1
    I1 = ICON (I+NEAN,1)
    I2 = ICON (I+NEAN,2)
    I3 = ICON (I+NEAN,3)
    I4 = ICON (I+NEAN,4)
    ICON (NE,1) = I1 + KNO2
    ICON (NE,2) = I2 + KNO2
    ICON (NE,3) = I3 + KNO2
 60 ICON (NE,4) = I4 + KNO2
C
    RETURN
    END
    SUBROUTINE BETPON (PI,XL,A,B,R,T,NPA,NBH,NO,NE,X,Y,Z,ICON)
C ---------------------------------------------------------------------
C
C   Objetivo : Calculo da malha das superficies entre os pontoons
C
C   Autor    : Carlos Augusto LEVI
C
C   Data     : Junho/1992
C
C ---------------------------------------------------------------------
C
    DIMENSION X(4000),Y(4000),Z(4000),ICON(4000,4)
C
C   COMPRIMENTO DO ANGULO TETA
C
    NOAN = NO
    ALFA = ASIN (B/2.0/R)
    TETA = PI/2.0 - 2.0* ALFA
C
    NPA1 = NPA + 1
    NBH1 = NBH + 1
C
C   COORDENADAS DOS PONTOS AO LONGO DO PERIMETRO
C
    DO 10 I = 1 , NBH1
    RI = FLOAT (I-1)
    DT = TETA * ABS ( COS(RI*PI/NBH).-1.0) / 2.0
C
C   COORDENADAS DOS PONTOS AO LONGO DA ALTURA
C
    DO 10 J = 1 , NPA1
    RJ = FLOAT (J-1)
    HH = A * ABS (COS(RJ*PI/NPA)-1.0) /2.0
C
C  COORDENADAS DOS NOS NO BURACO ENTRE PONTOONS NO REFERENCIAL GLOBAL
C
    NO = NO + 1
    X(NO) = XL - R * SIN (ALFA+DT)
    Y(NO) = XL - R * COS (ALFA+DT)
    Z(NO) = - T + HH
C
 10 CONTINUE
C
```

```
C   CONECTIVIDADE DOS ELEMENTOS
C
    DO 20 I = 1 , NBH
      IR1 = (I-1) * NPA1 + NOAN
      IR2 =  I  * NPA1 + NOAN
    DO 20 J = 1 , NPA
C
      NE = NE + 1
      ICON (NE,1) = IR1 + J
      ICON (NE,2) = IR1 + J + 1
      ICON (NE,3) = IR2 + J + 1
      ICON (NE,4) = IR2 + J
C
 20 CONTINUE
C
    RETURN
    END
    SUBROUTINE BETCON (PI,XL,A,B,R,T,NPA,NBHL,NO,NE,X,Y,Z,ICON)
C ---------------------------------------------------------------------
C
C   Objetivo : Calculo da malha da superficie complementar
C
C   Autor    : Carlos Augusto LEVI
C
C   Data     : Junho/1992
C
C ---------------------------------------------------------------------
C
    DIMENSION X(4000),Y(4000),Z(4000),ICON(4000,4)
C
C   COMPRIMENTO DO ANGULO TETA
C
    NOAN = NO
    ALFA = ASIN (B/2.0/R)
    TETA = 3.0*PI/2.0 - 2.0* ALFA
C
    NPA1  = NPA  + 1
    NBHL1 = NBHL + 1
C
C   COORDENADAS DOS PONTOS AO LONGO DO PERIMETRO
C
    DO 10 I = 1 , NBHL1
    RI = FLOAT (I-1)
    DT = TETA * ABS ( COS(RI*PI/NBHL) -1.0) / 2.0
C
C   COORDENADAS DOS PONTOS AO LONGO DA ALTURA
C
    DO 10 J = 1 , NPA1
    RJ = FLOAT (J-1)
    HH = A * ABS (COS(RJ*PI/NPA)-1.0) /2.0
C
C   COORDENADAS DOS NOS DO COMPLEMENTO NO REFERENCIAL GLOBAL
C
    NO = NO + 1
    X(NO) = XL + R * SIN (ALFA+DT)
    Y(NO) = XL - R * COS (ALFA+DT)
    Z(NO) = - T + HH
C
 10 CONTINUE
C
C   CONECTIVIDADE DOS ELEMENTOS
C
    DO 20 I = 1 , NBHL
      IR1 = (I-1) * NPA1 + NOAN
      IR2 =  I  * NPA1 + NOAN
    DO 20 J = 1 , NPA
C
      NE = NE + 1
      ICON (NE,1) = IR1 + J
      ICON (NE,2) = IR2 + J
      ICON (NE,3) = IR2 + J + 1
```

```
      ICON (NE,4) = IR1 + J + 1
C
 20  CONTINUE
C
    RETURN
    END
    SUBROUTINE COLCYL(PI,XL,A,B,R,T,NPL,NBHL,NBH,NCR,NO,NE,X,Y,Z,ICON)
C -------------------------------------------------------------------
C
C    Objetivo : Calculo da coluna cilindrica
C
C    Autor    : Carlos Augusto LEVI
C
C    Data     : Junho/1992
C
C -------------------------------------------------------------------
C
    DIMENSION X(4000),Y(4000),Z(4000),ICON(4000,4)
C
C    DADOS GERAIS
C
    NOAN  = NO
    NBH1  = NBH  + 1
    NBHL1 = NBHL + 1
    NPL1  = NPL  + 1
    NCR1  = NCR  + 1
C
    ALFA  = ASIN (B/2.0/R)
    TETA  = PI/2.0 - 2.0* ALFA
    TETAL = 3.0*PI/2.0 - 2.0* ALFA
C
    DD = T - A
C
C    CALCULO DAS COORDENADAS ACIMA DO PONTOON 1
C
    DO 10 I = 1 , NPL1
       RI = FLOAT (I-1)
       XX = B * ABS (COS(RI*PI/NPL) -1.0) / 2.0
       XR = -B/2.0 + XX
       YR = SQRT (R*R - XR*XR)
    DO 10 J = 1 , NCR1
       RJ = FLOAT (J-1)
       DZ = DD * ABS ( COS(RJ*PI/NCR) -1.0) / 2.0
C
       NO = NO + 1
       X(NO) = XL + XR
       Y(NO) = XL - YR
       Z(NO) = - T + A + DZ
 10  CONTINUE
C
C    INCIDENCIAS ACIMA DO PONTOON 1
C
    DO 15 I = 1 , NPL
      IR1 = (I-1) * NCR1 + NOAN
      IR2 =  I  * NCR1 + NOAN
    DO 15 J = 1 , NCR
      NE = NE + 1
      ICON (NE,1) = IR1 + J
      ICON (NE,2) = IR2 + J
      ICON (NE,3) = IR2 + J + 1
      ICON (NE,4) = IR1 + J + 1
 15  CONTINUE
C
C    CALCULO DAS COORDENADAS ACIMA DO COMPLEMENTO
C
    NOAN = NO
C
    DO 22 I = 1 , NBHL1
       RI = FLOAT (I-1)
       DT = TETAL* ABS ( COS(RI*PI/NBHL) -1.0) / 2.0
       DO 20 J = 1 , NCR1
```

```
          RJ = FLOAT (J-1)
          DZ = DD * ABS ( COS(RJ*PI/NCR) -1.0) / 2.0
C
          NO = NO + 1
          X(NO) = XL + R * SIN (ALFA+DT)
          Y(NO) = XL - R * COS (ALFA+DT)
          Z(NO) = - T + A + DZ
 20     CONTINUE
 22  CONTINUE
C
C    INCIDENCIAS ACIMA DO COMPLEMENTO
C
     DO 25 I = 1 , NBHL
       IR1 = (I-1) * NCR1 + NOAN
       IR2 =  I  * NCR1 + NOAN
     DO 25 J = 1 , NCR
       NE = NE + 1
       ICON (NE,1) = IR1 + J
       ICON (NE,2) = IR2 + J
       ICON (NE,3) = IR2 + J + 1
       ICON (NE,4) = IR1 + J + 1
 25  CONTINUE
C
C    CALCULO DAS COORDENADAS ACIMA DO PONTOON 2
C
     NOAN = NO
C
     DO 30 I = 1 , NPL1
        RI = FLOAT (I-1)
        YY = B * ABS (COS(RI*PI/NPL) -1.0) / 2.0
        YR =  B/2.0 - YY
        XR = SQRT (R*R - YR*YR)
     DO 30 J = 1 , NCR1
        RJ = FLOAT (J-1)
        DZ = DD * ABS ( COS(RJ*PI/NCR) -1.0) / 2.0
C
        NO = NO + 1
        X(NO) = XL - XR
        Y(NO) = XL + YR
        Z(NO) = - T + A + DZ
 30  CONTINUE
C
C    INCIDENCIAS ACIMA DO PONTOON 2
C
     DO 35 I = 1 , NPL
       IR1 = (I-1) * NCR1 + NOAN
       IR2 =  I  * NCR1 + NOAN
     DO 35 J = 1 , NCR
       NE = NE + 1
       ICON (NE,1) = IR1 + J
       ICON (NE,2) = IR2 + J
       ICON (NE,3) = IR2 + J + 1
       ICON (NE,4) = IR1 + J + 1
 35  CONTINUE
C
C    CALCULO DAS COORDENADAS ACIMA DO BURACO ENTRE PONTOONS
C
     NOAN = NO
C
     DO 40 I = 1 , NBH1
        RI = FLOAT (I-1)
        DT = TETA * ABS ( COS(RI*PI/NBH) -1.0) / 2.0
     DO 40 J = 1 , NCR1
        RJ = FLOAT (J-1)
        DZ = DD * ABS ( COS(RJ*PI/NCR) -1.0) / 2.0
C
        NO = NO + 1
        X(NO) = XL - R * SIN (ALFA+DT)
        Y(NO) = XL - R * COS (ALFA+DT)
        Z(NO) = - T + A + DZ
 40  CONTINUE
```

```
C
C   INCIDENCIAS ACIMA DO PONTOON 2
C
    DO 45 I = 1 , NBH
      IR1 = (I-1) * NCR1 + NOAN
      IR2 =  I  * NCR1 + NOAN
    DO 45 J = 1 , NCR
      NE = NE + 1
      ICON (NE,1) = IR1 + J
      ICON (NE,2) = IR1 + J + 1
      ICON (NE,3) = IR2 + J + 1
      ICON (NE,4) = IR2 + J
 45 CONTINUE
C
    RETURN
    END
    SUBROUTINE BASCOL(PI,XL,B,R,T,NFR,NPL,NBHL,NBH,NO,NE,X,Y,Z,ICON)
C ------------------------------------------------------------------
C
C   Objetivo : Calculo da base da coluna
C
C   Autor   : Carlos Augusto LEVI
C
C   Data    : Junho/1992
C
C ------------------------------------------------------------------
C
    DIMENSION X(4000),Y(4000),Z(4000),ICON(4000,4)
C
C   DADOS GERAIS
C
    NOAN  = NO
    NBH1  = NBH + 1
    NBHL1 = NBHL + 1
    NPL1  = NPL + 1
    NFR1  = NFR + 1
C
    ALFA = ASIN (B/2.0/R)
    TETA = PI/2.0 - 2.0* ALFA
    TETAL = 3.0*PI/2.0 - 2.0* ALFA
C
C   CALCULO DAS COORDENADAS NO PERIMETRO DO PONTOON 1
C
    DO 10 I = 1 , NPL1
      RI = FLOAT (I-1)
      XX = B * ABS (COS(RI*PI/NPL) -1.0) / 2.0
      XR = -B/2.0 + XX
      YR = SQRT (R*R - XR*XR)
      CA = XR / R
      SA = YR / R
    DO 10 J = 1 , NFR1
      RJ = FLOAT (J-1)
      DR = R * ABS ( COS(RJ*PI/NFR/2.0) )
C
      NO = NO + 1
      X(NO) = XL + DR * CA
      Y(NO) = XL - DR * SA
      Z(NO) = - T
 10 CONTINUE
C
C   INCIDENCIAS ACIMA DO PONTOON 1
C
    DO 15 I = 1 , NPL
      IR1 = (I-1) * NFR1 + NOAN
      IR2 =  I  * NFR1 + NOAN
    DO 15 J = 1 , NFR
      NE = NE + 1
      ICON (NE,1) = IR1 + J
      ICON (NE,2) = IR1 + J + 1
      ICON (NE,3) = IR2 + J + 1
      ICON (NE,4) = IR2 + J
```

```
 15  CONTINUE
C
C    CALCULO DAS COORDENADAS NO PERIMETRO DO COMPLEMENTO
C
     NOAN = NO
C
     DO 20 I = 1 , NBHL1
        RI = FLOAT (I-1)
        DT = TETAL* ABS ( COS(RI*PI/NBHL) -1.0) / 2.0
     DO 20 J = 1 , NFR1
        RJ = FLOAT (J-1)
        DR = R * ABS ( COS(RJ*PI/NFR/2.0) )
C
        NO = NO + 1
        X(NO) = XL + DR * SIN (ALFA+DT)
        Y(NO) = XL - DR * COS (ALFA+DT)
        Z(NO) = - T
 20  CONTINUE
C
C    INCIDENCIAS NO PERIMETRO DO COMPLEMENTO
C
     DO 25 I = 1 , NBHL
        IR1 = (I-1) * NFR1 + NOAN
        IR2 =  I  * NFR1 + NOAN
     DO 25 J = 1 , NFR
        NE = NE + 1
        ICON (NE,1) = IR1 + J
        ICON (NE,2) = IR1 + J + 1
        ICON (NE,3) = IR2 + J + 1
        ICON (NE,4) = IR2 + J
 25  CONTINUE
C
C    CALCULO DAS COORDENADAS NO PERIMETRO DO PONTOON 2
C
     NOAN = NO
C
     DO 30 I = 1 , NPL1
        RI = FLOAT (I-1)
        YY = B * ABS (COS(RI*PI/NPL) -1.0) / 2.0
        YR =  B/2.0 - YY
        XR = SQRT (R*R - YR*YR)
        CA = XR / R
        SA = YR / R
     DO 30 J = 1 , NFR1
        RJ = FLOAT (J-1)
        DR = R * ABS ( COS(RJ*PI/NFR/2.0) )
C
        NO = NO + 1
        X(NO) = XL - DR * CA
        Y(NO) = XL + DR * SA
        Z(NO) = - T
 30  CONTINUE
C
C    INCIDENCIAS NO PERIMETRO DO PONTOON 2
C
     DO 35 I = 1 , NPL
        IR1 = (I-1) * NFR1 + NOAN
        IR2 =  I  * NFR1 + NOAN
     DO 35 J = 1 , NFR
        NE = NE + 1
        ICON (NE,1) = IR1 + J
        ICON (NE,2) = IR1 + J + 1
        ICON (NE,3) = IR2 + J + 1
        ICON (NE,4) = IR2 + J
 35  CONTINUE
C
C    CALCULO DAS COORDENADAS NO PERIMETRO DO BURACO ENTRE PONTOONS
C
     NOAN = NO
C
     DO 40 I = 1 , NBH1
```

```
      RI = FLOAT (I-1)
      DT = TETA * ABS ( COS(RI*PI/NBH) -1.0) / 2.0
    DO 40 J = 1 , NFR1
      RJ = FLOAT (J-1)
      DR = R * ABS ( COS(RJ*PI/NFR/2.0) )
C
      NO = NO + 1
      X(NO) = XL - DR * SIN (ALFA+DT)
      Y(NO) = XL - DR * COS (ALFA+DT)
      Z(NO) = - T
 40 CONTINUE
C
C   INCIDENCIAS NO PERIMETRO DO BURACO ENTRE PONTOONS
C
    DO 45 I = 1 , NBH
      IR1 = (I-1) * NFR1 + NOAN
      IR2 =  I  * NFR1 + NOAN
    DO 45 J = 1 , NFR
      NE = NE + 1
      ICON (NE,1) = IR1 + J
      ICON (NE,2) = IR2 + J
      ICON (NE,3) = IR2 + J + 1
      ICON (NE,4) = IR1 + J + 1
 45 CONTINUE
C
    RETURN
    END
```

APÊNDICE F - Padronização dos nomes dos nós

F.1. Apresentação

Este apêndice se destina a explicar as regras de padronização utilizadas pelo ACF para criação automática dos nomes dos nós do hiperdocumento.

É apresentada a regra para cada tipo de nó criado, em ordem alfabética.

A sintaxe utilizada é a seguinte:

- o sinal "+" representa concatenação;
- palavras ou letras maiúsculas em negrito são efetivamente utilizadas;
- caracteres especiais em negrito são efetivamente utilizados;
- frases minúsculas explicam qual conteúdo será concatenado;

F.2. Padronização

Alterações: **A** + nome do módulo

Ex: APOTOBA, AMESH

Comentário: **C** + número sequencial do comentário + nome do módulo

Ex: C1BASCOL, C2MESH

Descrição_oper: Se common - **$** + **&** + nome do common
Se variável - **$** + nome da variável + letra sequencial
Se arquivo - **$** + **&** + número do arquivo

Ex: $&DADOS1, $NPL01, $&5

Documentação: **D** + nome do módulo

Ex: DMESH, DPOTOBA

Intermediário: **I** + *número* + nome do módulo

Onde *número* vale:

1 - intermediário para o nó do módulo principal.
2 - igual ao 1, contendo botões para os nós de versão e alterações.
3 - intermediário para os nós dos outros módulos, em sua chamada.
4 - igual ao 3, contendo botões para os nós de versão e alterações.
5 - intermediário para os nós dos outros módulos, no nó tipo Módulo.
6 - igual ao 5, contendo botões para os nós de versão e alterações.

Ex: I1MESH, I4POTOBA

Módulo: nome do módulo

Ex: MESH, POTOBA

Observações: **O** + **-** + nome do módulo

Ex: O-MESH

Operando: Se variável - nome da variável + número sequencial de um módulo
Se common - **&** + nome do common
Se arquivo - **ARQ** + número do arquivo
Se conteúdo do arquivo - **_** + nome externo do arquivo

Ex: NPL01, &DADOS1, ARQ5, _RESUL

Relatório: **RELQUAL** - Se Relatório de Avaliação da Qualidade
RELCOMP - Se Relatório de Avaliação da Complexidade

Versão: **V** + número da versão + nome do módulo

Ex: V1POTOBA, V2MESH