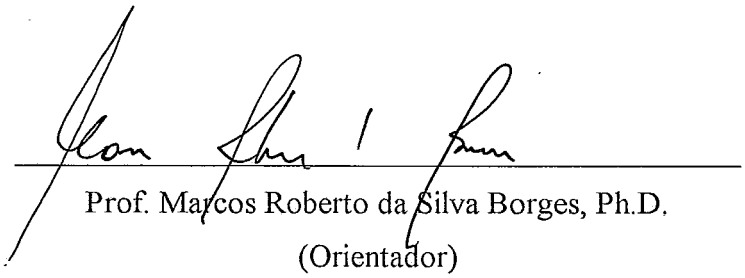
Recuperação Cooperativa do Projeto Arquitetônico de Sistemas

Maria Cláudia Reis Cavalcanti

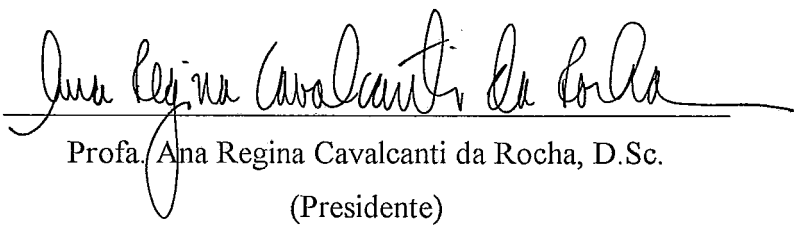
TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO EM ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:



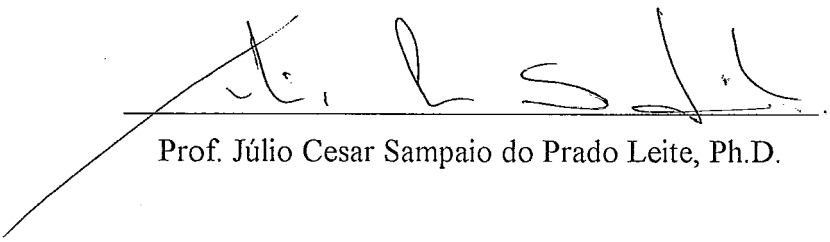
Prof. Marcos Roberto da Silva Borges, Ph.D.

(Orientador)



Profa. Ana Regina Cavalcanti da Rocha, D.Sc.

(Presidente)



Prof. Júlio Cesar Sampaio do Prado Leite, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

OUTUBRO DE 1994

CAVALCANTI, MARIA CLÁUDIA REIS
Recuperação Cooperativa do Projeto Arquitetônico de Sistemas [Rio de Janeiro] 1994.
VIII, 84 págs., 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1994)
Tese - Universidade Federal do Rio de Janeiro, COPPE
1. Engenharia de Software 2. Engenharia Reversa
3. Hipertexto
4. Trabalho Cooperativo Suportado por Computador
I. COPPE/UFRJ II. Título (série).

Agradecimentos

Ao meu orientador e amigo Marcos Borges, que com a sua confiança na minha capacidade, incentivou-me a iniciar o curso de Mestrado, e cuja força, clareza e objetividade guiaram-me durante todo o desenvolvimento deste trabalho.

Aos professores Ana Regina C. da Rocha e Júlio Leite, que se prontificaram a integrar a banca examinadora.

Ao amigo Fernando Manso, pela orientação que guiou meus primeiros passos na escolha do tema de tese.

Aos professores da COPPE/UFRJ, cujos ensinamentos tornaram possível o alcance desta meta.

Ao NCE/UFRJ que viabilizou a obtenção dos créditos necessários e permitiu a utilização de seus computadores, programas e biblioteca, para a realização dos trabalhos e pesquisas.

Ao CNPq, pelo apoio financeiro.

Às bibliotecárias do NCE/UFRJ, pela ajuda nas pesquisas realizadas.

À secretaria da COPPE Sistemas, por sua eficiência e prontidão, e ao Centro de Documentação, que com seu acervo enriqueceu minhas pesquisas.

Aos amigos da sala E-1040 do NCE/UFRJ, Claudinha, Jonathan e Maria Luíza, por sua compreensão e apoio durante estes dois últimos anos de convivência.

Aos colegas do NCE/UFRJ, que contribuíram com seu incentivo, críticas e opiniões, em especial a Manoel Lois e Carlo Emanoel, analistas responsáveis pelo sistema TEDMOS, pelo incentivo e apoio durante o experimento realizado.

À minha grande amiga Renata, companheiríssima nos bons e maus momentos do curso, que contribuiu com valiosas sugestões, e que hoje compartilha comigo a saudade do nosso grupo de estudo com a amiga Luciana.

Aos meus amigos, que me acompanharam e incentivaram nesta jornada, e de cujas vidas subtraí, em parte, minha presença e minha atenção.

A meus pais por toda uma vida de apoio e carinho, e a minha irmã pela amizade, incentivo e por suas valiosas opiniões sobre o texto.

Finalmente, a todos aqueles que direta ou indiretamente contribuíram para a conclusão deste trabalho, cuja lembrança me falta neste momento.

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

Recuperação Cooperativa do Projeto Arquitetônico de Sistemas

Maria Cláudia Reis Cavalcanti

Outubro, 1994

Orientador: Prof. Marcos Roberto da Silva Borges
Programa: Engenharia de Sistemas e Computação

Esta tese propõe um ambiente cooperativo para apoiar o processo de Engenharia Reversa, mais especificamente a Recuperação do Projeto Arquitetônico de sistemas. Diferentemente da maioria das ferramentas existentes para apoiar este processo, o ambiente proposto complementa a extração automática da estrutura modular e da árvore conceitual de um sistema, através da captura do conhecimento de seus especialistas. Para isso, utiliza-se um modelo de argumentação, através do qual os especialistas interajem entre si, complementando a informação extraída automaticamente. Ao final do processo, são gerados dois documentos, o Projeto Arquitetônico do sistema e um registro do que foi discutido entre os especialistas. Um protótipo denominado ARCoPAS, foi construído para permitir a experimentação do ambiente proposto.

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

Cooperative Recovery of Systems Architectural Design

Maria Cláudia Reis Cavalcanti

October, 1994

Thesis Supervisor: Prof. Marcos Roberto da Silva Borges

Department: Systems Engineering and Computer Science

This thesis presents an environment to support the Reverse Engineering process, more specifically the Architectural Design Recovery of systems. Differently from most of the existing Reverse Engineering tools, the proposed environment complements the automatic extraction of the system modular structure and conceptual tree, with the acquisition of the especialists knowledge. Through the use of an argumentation model, the especialists cooperate with each other, completing the information extracted automatically. At the end of the process, two documents are produced, the Architectural Design of the system, and the record of the discussions that took place between the especialists. A prototype called ARCoPAS, was implemented to allow experimentations of the proposed environment.

Índice

Capítulo 1
Introdução

A fase de manutenção de software apresenta altos custos, decorrentes do trabalho com o software sem documentação. Para atacar este problema alguns investem no processo de desenvolvimento de software, utilizando metodologias que apoiam todo o processo e que geram uma documentação adequada. Outros investem no processo reverso, isto é, na extração de documentação a partir do software já desenvolvido.

A realidade de 80% dos softwares existentes consumirem grandes recursos em sua manutenção [Lafu90] exige uma atitude imediata. A solução adotada por aqueles que investem no processo de desenvolvimento, geralmente envolve a construção de um novo sistema e o descarte do sistema antigo. Entretanto, a atual valorização do reaproveitamento, tem conquistado cada vez mais adeptos à corrente que investe no processo reverso.

As ferramentas desenvolvidas até hoje para apoiar a Engenharia Reversa, pouco investem na participação humana. A maioria destas ferramentas são analisadores estáticos automáticos [Lafu90] que geram relatórios, como por exemplo o fluxograma ou a estrutura modular dos programas. No entanto, os especialistas no sistema que se quer "reverter", detém informações muito enriquecedoras que estas ferramentas totalmente automáticas não conseguem extrair. Assim sendo, faz-se necessário o investimento em ferramentas que possibilitem a interação com o conhecimento humano.

O conhecimento sobre um sistema encontra-se disperso entre os especialistas, que em geral são os membros da equipe que o mantém. Para extrair este conhecimento é preciso que a ferramenta de apoio ofereça mecanismos de captura incremental. Isto é, à medida em que o conhecimento for capturado, este deve ser organizado e armazenado, de forma que os especialistas tenham acesso à informação acumulada até o momento. A fácil visualização desta informação é muito importante para que o processo de captura tenha sucesso. Além disso, a ferramenta de apoio deve permitir a interação entre os especialistas, de modo a promover a cooperação entre eles, e incentivar a captura de todo o conhecimento sobre o sistema.

O objetivo deste trabalho é investir no reaproveitamento do software existente, contando com a participação da equipe de manutenção do mesmo. Mais especificamente, propõe-se um ambiente de suporte à Recuperação Cooperativa do Projeto Arquitetônico de Sistemas. Este ambiente pressupõe a divisão da atividade de

Recuperação de Projeto em duas etapas, uma automática e outra cooperativa. Na etapa automática extrai-se o que for possível do código fonte, através da utilização de uma ferramenta de análise de códigos. Na etapa cooperativa, a informação extraída automaticamente, é complementada pelos especialistas no sistema, apoiados por ferramentas que implementam tecnologias de hipertexto e trabalho cooperativo.

Ao final do processo de recuperação, obtém-se dois documentos: o Projeto Arquitetônico propriamente dito, e a Discussão sobre a complementação deste. Para representar o Projeto Arquitetônico, propõe-se a utilização da notação de Projeto HTDN, que corresponde a uma adaptação da notação TDN [Guez91] sob a forma de hipertexto. O documento da Discussão contém a interação entre os especialistas durante a etapa cooperativa. Para a representação desta informação propõe-se a utilização do modelo de argumentação IBIS.

Para experimentar o ambiente proposto implementamos um protótipo denominado ARCoPAS, que integra duas ferramentas harmonicamente: ExEM e FolioViews. A primeira delas apóia a etapa automática da Recuperação de Projeto, enquanto que a segunda apóia a etapa cooperativa. A partir da utilização do ARCoPAS em um sistema real, foi possível identificar melhorias e planejar a construção de um ambiente ainda mais completo.

Além da introdução e conclusão, este trabalho está organizado em mais 6 capítulos. No segundo capítulo apresenta-se a fase de manutenção e seus problemas, que foram a motivação para o desenvolvimento deste trabalho. Ainda neste capítulo, discute-se também, algumas das abordagens utilizadas para amenizar estes problemas, enfatizando a abordagem escolhida. No terceiro e quarto capítulos apresenta-se um resumo das tecnologias de hipertexto e trabalho cooperativo suportado por computador, nas quais esta proposta foi baseada. O quinto capítulo descreve a proposta de um ambiente para Recuperação Cooperativa do Projeto Arquitetônico. Finalmente, o protótipo ARCoPAS, desenvolvido para implementar este ambiente, é descrito e avaliado, nos capítulos 6 e 7, respectivamente.

Capítulo 2
A Fase de Manutenção

A manutenção é a última fase do ciclo de vida clássico de software, e inicia-se quando o software é colocado em operação junto a seus usuários. A principal característica da manutenção é o fato de constituir a maior parte da vida útil de um software, estendendo-se até que o mesmo seja desativado. A maioria dos autores cita estudos que mostram que a fase de manutenção pode consumir mais de 60% [Zelk79, Ghez91] dos custos previstos para o ciclo de vida de um software. Alguns estudos mais recentes comprovam que a manutenção pode consumir até 90% dos custos [Gorl91].

A manutenção apresenta-se aparentemente tranquila, fazendo com que os engenheiros de software incorram no erro de subestimá-la. Entretanto, atualmente, esta fase já é conhecida como a mais crítica, comparada até mesmo a um "iceberg" [Pres87]. Assim, o desenvolvimento de um software deve ter presente em cada fase a preocupação com facilitar a sua manutenção, através de uma série de medidas, incluindo o estabelecimento de diretrizes e a utilização de padrões.

Podemos identificar quatro tipos de manutenção de software: corretiva, adaptativa, aperfeiçoadora e preventiva [Simo87, Benn91, Pres87]. A manutenção ***corretiva*** é aquela que se presta a corrigir os erros que não foram identificados durante a fase de testes do software. A ***adaptativa*** acontece quando uma mudança no ambiente afeta o funcionamento do software, como por exemplo, uma troca de máquina, a aquisição de um novo periférico, etc... A manutenção ***aperfeiçoadora***, que possui maior incidência (entre 50% e 75% [Simo87, Ghez91, Benn91]) tem o propósito de realçar e trazer melhorias para o software, valorizando-o mais em relação ao mercado, ou atendendo a novos requisitos do usuário.

A manutenção ***preventiva*** não é identificada por todos os autores, e é a que tem menos atenção dos responsáveis pela fase de manutenção. Este tipo de manutenção tem o objetivo de preparar o software para futuras manutenções, isto é, investir na redução dos custos com os demais tipos de manutenção, melhorando a qualidade dos programas, principalmente em relação a sua manutenibilidade. Todo software cujo desenvolvimento tenha ocorrido sem a utilização de métodos adequados, é um ótimo candidato para a manutenção preventiva.

2.1. Os Problemas da fase de manutenção

Uma vez que facilitar a manutenção de um software - manutenibilidade - deve ser o objetivo das fases de desenvolvimento, as deficiências ocorridas nestas fases é que levam aos principais problemas da fase de manutenção: ilegibilidade, descaracterização, obsolescência, degradação e erros críticos nos programas.

A manutenibilidade de um programa depende de sua legibilidade. Um programa legível é aquele que é claro, conciso, estilizado e modular. *Claro* por que implementa algorítmos simples e de fácil entendimento; *conciso* por que implementa o mínimo de código necessário para executar a tarefa a que se destina; *estilizado* por que utiliza um formato padrão, endentações e comentários; e *modular* por que seu código está disposto em uma estrutura de módulos altamente independentes entre si [Roch87].

Os programas ilegíveis exigem um grande esforço para que um programador possa entendê-los e efetuar a sua manutenção. Além disso, nem sempre se pode contar com os autores de um programa deste tipo, para auxiliar no processo de entendimento. Quanto mais antigo for o programa, provavelmente maior é o grau de ilegibilidade do mesmo.

A inexistência de uma documentação adequada do software, contando a história de seu desenvolvimento e identificando seus objetivos, pode acarretar a descaracterização do mesmo. A equipe de manutenção, por desconhecer o software em alteração, pode cometer erros de interpretação quanto as suas características básicas, provocando a perda de identidade do mesmo. Os métodos de engenharia de software primam pela geração de uma farta documentação, que tenta garantir a caracterização de um software.

A todo momento estão surgindo tecnologias avançadas, tanto em relação a hardware quanto a software. As empresas de software vêem-se obrigadas a incorporar estas inovações tecnológicas, recém saídas dos laboratórios de pesquisa, para manter seu produto a frente do mercado. No entanto, o alto custo de manutenção de um software muitas vezes evita que se implemente estas melhorias no mesmo. Um software que não investe na compatibilidade com novas máquinas, sistemas operacionais ou redes, torna-se rapidamente obsoleto.

Durante a manutenção de um software, todas as fases do desenvolvimento são revividas para a parte do software que sofrerá mudanças. Entretanto, a facilidade de acesso à alteração de um programa, isto é, a sua maleabilidade, e a pressão exercida

pelos usuários, leva os profissionais do ramo a negligenciar o trabalho de manutenção e não cumprir a risca os procedimentos que visam facilitar manutenções futuras. Portanto, mesmo para um software desenvolvido de forma criteriosa, a fase de manutenção pode degradá-lo a tal ponto, que se torna imprescindível substituí-lo.

Além de todos os problemas apresentados, a manutenção pode ainda introduzir erros não interceptados pelos testes. Nem sempre estes testes abrangem todo o programa, detendo-se apenas nas partes alteradas do mesmo, ou ainda, nem sempre é possível verificar completamente o programa. Este talvez seja o problema mais crítico da fase de manutenção, uma vez que os erros podem ter consequências catastróficas.

2.2. Enfrentando os problemas

Entre os esforços no sentido de enfrentar os problemas comentados anteriormente, podemos identificar duas abordagens principais: metodologias de desenvolvimento de software e Engenharia Reversa. Ambas as abordagens contam com diversas ferramentas de apoio. Entre as ferramentas que apoiam as metodologias de desenvolvimento de software, existem aquelas que abrangem desde a fase de análise até a implementação, com geração automática de código fonte, mas muitas delas restringem-se a apoiar apenas parte do processo de desenvolvimento.

Os métodos de desenvolvimento de software podem ser classificados em três tipos, segundo a sua orientação: processos, dados e objetos [Somm85, Booc94]. Métodos orientados a processos foram propostos por diversos autores, tais como DeMarco [DeMa79], Gane e Sarson [GaSa79], e Yourdon e Constantine [YoCo79, Your79]. A Análise Estruturada e o Projeto Estruturado são exemplos deste tipo de método, e fazem parte de uma metodologia que guia o desenvolvimento de software desde a fase de análise até a sua implementação [Guez91]. A ferramenta Cradle® da Yourdon Inc., foi construída para suportar o método estruturado Yourdon no desenvolvimento de sistemas de informação e tempo-real [Oman90].

O JSD ("Jackson's Systems Development") é um método orientado a dados, que foi proposto por Jackson em 1983, e tem sido largamente utilizado desde então [Came88, PrLuLe91]. Este método é uma extensão do método JSP ("Jackson Structured Programming"), que abrange todas as fases do desenvolvimento de sistemas [Guez91]. Para apoiar os métodos JSD/JSP de Jackson foi desenvolvido um pacote de ferramentas: Speedbuilder, JSP Tool, Adacode e JSP Workbench [Oman90].

Em resposta à crescente complexidade dos sistemas e disponibilidade de máquinas poderosas, surgiram métodos orientados a objeto, como os propostos por Booch [Booc86] e Coad/Yourdon [CoYo92, CoYo93]. A ferramenta OOATool® apóia o método de Análise Baseada em Objetos de Coad/Yourdon [CoYo92], enquanto a ferramenta Rational Rose® apóia o método de Booch [Whit94].

Atualmente, existem softwares em operação, cujo desenvolvimento não se baseou em uma metodologia de desenvolvimento de software. Mesmo com a proliferação de ferramentas CASE no mercado, visando auxiliar as fases de desenvolvimento, a maioria dos softwares existentes são antigos, com idade variando entre 10 e 20 anos [Benn91, Your89a], e carentes de documentação. Em geral, estes softwares foram desenvolvidos sem a utilização de um método específico, não possuem manuais associados, e consequentemente, são de difícil manutenção. Para viabilizar a sua manutenção, é necessário investir na manutenção preventiva dos mesmos. Os esforços neste sentido quase sempre envolvem o emprego da Engenharia Reversa.

2.2.1. Engenharia Reversa

A Engenharia Reversa é uma metodologia que se caracteriza por atender tanto aos envolvidos com o desenvolvimento de software, quanto àqueles envolvidos com a sua manutenção. Devido ao fato de ser uma tecnologia muito recente, existe ainda muita confusão em relação à terminologia empregada. Até mesmo o termo "engenharia reversa" tem um uso controvertido.

Tomando como base o artigo de Chikofsky e Cross [ChCr90], vamos definir a seguir o significado dos seguintes termos: engenharia reversa, redocumentação, recuperação de projeto, engenharia progressiva[1], reestruturação e reengenharia. Esta definição tem o objetivo de permitir o entendimento correto dos termos acima, que serão citados eventualmente no decorrer do texto.

O termo *engenharia reversa* já vem sendo usado na engenharia eletrônica, para designar o processo de análise de um sistema de hardware, com o propósito de gerar um clone do mesmo. Analogamente, na engenharia de software, este termo é usado para designar o processo de análise de um sistema de software, entretanto seu propósito é diferente. Em momento algum a "engenharia reversa" altera ou modifica o sistema de software, pelo contrário, sua intensão é analisá-lo, identificando seus

[1] O termo engenharia progressiva é usado como tradução para o termo em inglês "forward engineering".

componentes e interrelacionamentos, para extrair informações mais abstratas sobre seu funcionamento, que irão facilitar o entendimento do mesmo. A engenharia reversa pode ser aplicada em qualquer fase do ciclo de vida, sendo possível a sua utilização mesmo para sistemas inacabados.

A *redocumentação* e a *recuperação de projeto* são duas subáreas da engenharia reversa. A primeira envolve a revisão de uma representação do sistema[2], gerando outra dentro do mesmo nível de abstração, isto é, através da melhoria do aspecto de uma representação, a redocumentação visa facilitar o seu entendimento. Quando a documentação não existe, cria-se uma a partir de um nível menos abstrato. Um exemplo de redocumentação é a transformação da própria listagem do código, através da utilização de ferramentas de formatação conhecidas como "pretty printers". Os visualizadores de código, também podem ser considerados ferramentas de redocumentação, na medida em que melhoram o aspecto e facilitam o entendimento do código de um programa.

Além da definição apresentada por Chikofsky e Cross, encontramos na literatura atual mais duas definições para a atividade de *recuperação de projeto*, provenientes de autores também reconhecidos na área. É interessante notar, que em todas as definições abaixo, identifica-se a necessidade de interação com o conhecimento de especialistas.

> *"A Recuperação de Projeto é um subconjunto da Engenharia Reversa no qual, conhecimento do domínio, informação externa, e deduções são adicionados às observações do sistema em questão para identificar abstrações de alto nível significativas entre aquelas obtidas diretamente pelo exame do sistema em si."* [ChCr90]

> *"A Recuperação de Projeto recria abstrações de projeto a partir de uma combinação de código, documentação de projeto existente (se disponível), experiência pessoal, e conhecimento geral sobre o problema e o domínio da aplicação."* [Bigg89]

> *"... um engenheiro que está recuperando o projeto de um sistema repete a seguinte sequência de passos: inspeciona o sistema, gera hipóteses sobre as funções dos diversos componentes do sistema, tenta verificar estas hipóteses, e finalmente adiciona novos fatos a especificação de projeto do sistema."* [Koza90]

---

[2] Uma representação do sistema é um documento do sistema, gerado por uma fase de desenvolvimento do mesmo. Alguns autores usam o termo "elemento de software".

O termo *engenharia progressiva* é usado para designar o processo tradicional de passagem de um nível mais alto de abstração, para um nível mais próximo a implementação. Uma vez que o termo engenharia de software não esclarece o sentido do desenvolvimento, o adjetivo "progressiva" é usado para distinguir claramente o sentido deste, como o contrário à engenharia reversa.

A *reestruturação* é a transformação de representações dentro do mesmo nível de abstração preservando o comportamento externo do sistema. Este termo costuma ser usado denotando a alteração de um código fonte para uma forma melhor estruturada, entretanto o seu significado é mais abrangente, estendendo-se ao modelo de dados, planejamento de projeto ou estruturas de requerimento.

A *reengenharia* é um processo que envolve alguma forma de engenharia reversa para atingir um nível mais abstrato do sistema, somada a alguma forma de engenharia progressiva ou reestruturação. Usando outras palavras, a *reengenharia* de um sistema é a sua reconstrução de uma nova maneira, levando em consideração o resultado de uma análise feita sobre o mesmo.

2.2.2. Apoiando a Engenharia Reversa

Já é possível encontrar no mercado diversas ferramentas de apoio a Engenharia Reversa e atividades afins. Existem ferramentas que visam a *reestruturação* automática de programas, como por exemplo o COBOL Structuring Facility da IBM, o Superstructure comercializado pela Computer Data Systems, o Retrofit da Catalyst/Peat Marwick e o RECODER da Language Technology Inc. [Gorl91, Your89a]. Em geral, estas ferramentas partem do código fonte não estruturado, gerando um código estruturado e funcionalmente equivalente ao anterior.

As ferramentas de reestruturação, entretanto , apresentam alguns contrapontos. São geralmente pouco flexíveis no que diz respeito ao formato do novo programa gerado, causando muita estranheza entre os programadores. Outro fator negativo é que um programa ruim, não deixa de ser ruim só por que tornou-se estruturado. Ou seja, a reestruturação não pode realizar mágicas.

Além disso, a reestruturação deve ser oportunamente aplicada, isto é, uma seleção prévia dos programas candidatos a reestruturação deve ser feita, utilizando-se critérios para priorização dos mesmos. Segundo Yourdon [Your89a], os primeiros programas

que devem passar pela reestruturação devem ser aqueles de menor importância estratégica, isto é, os que oferecem menor risco se algo sair errado; em segundo, aqueles cuja confiabilidade é baixa, principalmente por parte do usuário; e em terceiro, aqueles cuja frequência de manutenção é alta, ou seja, cujos custos de manutenção são mais expressivos.

Para auxiliar no processo de seleção dos programas candidatos a reestruturação, também já existem ferramentas apropriadas. São as ferramentas métricas, que medem a qualidade dos programas, como por exemplo, o INSPECTOR da Language Technology Inc. e o Battle Map/Act da McCabe & Associates. As duas primeiras fornecem a complexidade ciclomática dos programas, que tem sido usada largamente para medir a confiabilidade e a manutenibilidade dos mesmos [Oman90]. Enquanto o INSPECTOR analisa apenas o código COBOL, o Battle Map/Act analisa, além do COBOL, códigos em C, FORTRAN, Ada, BASIC, PL/I, PASCAL e o Assembler dos processadores 8086 e 6502.

O processo de entendimento de programas, facilitado ou não pela reestruturação, continua presente na fase de manutenção, tomando em torno de 47% do tempo total [Gorl91]. Para agilizar este processo, entram em cena as ferramentas de redocumentação, que auxiliam a compreensão de programas, como os analisadores estáticos e visualizadores de código. Algumas destas ferramentas estão relacionadas a seguir [Your89a, Oman90, Gorl91].

- **VIA/Insight** da VIASOFT é um analisador estático e interativo de programas COBOL;
- **Objective C Browser** da Stepstone é um analisador estático de código C;
- **Vifor** da Software Tools and Technologies é um visualizador gráfico de código Fortran;
- **EDSA** ou Expert Dataflow Static Analysis da Array Systems Computing, é um analisador estático que permite visualizar somente as partes de programas Ada que interessam, utilizando a técnica de fatiamento[3] de programas;
- **Surgeon's Assistent** do Loyola College e University of Maryland é o protótipo de um editor de programas em C utilizando a técnica de fatiamento por decomposição[4];
- **Scan/Cobol** é um analisador estático de código Cobol;

[3] O fatiamento de programas é a tradução da expressão "Program slicing", que segundo Gallagher e Lyle [GaLy91], é uma técnica para restringir o comportamento de um programa a um subconjunto de interesse.

[4] Fatiamento por decomposição consiste em dividir o programa em dois ou mais componentes, chamados fatias.

- **Dependency Analysis Tool Set** da University of West Florida é um analisador de dependências para grandes sistemas em código C, que apresenta um grafo em que as entidades (variáveis de um programa C) são nós e as dependências são arcos entre estas;

Um aspecto importante para o entendimento de um software e a sua consequente manutenção, é a atualização da documentação dos programas. Como já foi dito, devido à pressão exercida sobre o programador de manutenção e à burocracia envolvida, a documentação dificilmente é atualizada. Para solucionar este problema, algumas das ferramentas citadas suportam a *redocumentação* de programas, gerando documentos, tais como diagramas da estrutura hierárquica, listas de referência cruzada, listagens formatadas ("pretty printers") e outros.

Entre as ferramentas que apoiam a *Recuperação de Projeto* podemos identificar aquelas que extraem o Projeto Detalhado, isto é, a especificação de cada módulo do sistema, e aquelas que extraem o Projeto Arquitetônico do sistema. A ferramenta Seela da Tuval Software Industries por exemplo, é uma ferramenta para a recuperação de projeto detalhado, pois apresenta sob a forma de uma linguagem de especificação de programas, o código estruturado escrito em Ada, COBOL, C, Pascal, PL/M ou Fortran [Oman90]. Já o sistema Rigi [MuTi92] traz para um ambiente de edição gráfica tri-dimensional, o Projeto Arquitetônico de um sistema escrito em código COBOL ou C.

A introdução de metodologias de desenvolvimento de software em ambientes carentes, pode ser feita naturalmente através da Engenharia Reversa. Em [LeFr91], Leite e Franco apresentam uma estratégia para recuperação de projeto de um sistema, gerando documentos no formato utilizado pelo método JSD. A partir destes documentos, o Reprojeto deste sistema é imediato, seguindo-se o método JSD em sua sequência natural. A experiência relatada no entanto, não contou com o apoio de ferramentas automáticas.

Recentemente, têm surgido ferramentas de apoio à *engenharia reversa* integradas a ferramentas de apoio à *engenharia progressiva*. Além de facilitar a introdução de metodologias de desenvolvimento, esta integração de ferramentas facilita também a *reengenharia* de sistemas. O sistema Re-Analyzer [OHTr94] é um exemplo de ferramenta integrada. Os documentos recuperados automaticamente pelo Re-Analyzer, são gerados no formato adequado a uma ferramenta CASE, que suporta o método de Análise Estruturada para sistemas de Tempo Real (SA/RT). Similarmente, o método RECAST ("Reverse Engineering CASe Technology") [EdMu93] apoia-se em ferramentas para viabilizar a engenharia reversa de sistemas, gerando documentos no

formato adequado às ferramentas que suportam o método de Análise e Projeto de Sistemas Estruturados (SSADM).

As ferramentas descritas anteriormente, são tentativas no sentido de minimizar os custos com a fase de manutenção, através do apoio à Engenharia Reversa. Este trabalho também investe na Engenharia Reversa, buscando o apoio de tecnologias de Hipertexto e Trabalho Cooperativo Suportado por Computador, para propor um ambiente adequado à Recuperação do Projeto Arquitetônico de sistemas.

Capítulo 3
Sistemas de Hipertexto

A capacidade de armazenamento dos computadores tem crescido incrivelmente em volume oferecido, e diminuído em espaço físico ocupado, viabilizando o armazenamento, em meio eletrônico, de informações volumosas, tais como imagens (fotos, gráficos e desenhos), animação (sequência de imagens) e som. A partir desta evolução tecnológica, foi possível implementar softwares conhecidos como *sistemas de hipermídia*, que utilizam todo o potencial conectivo e de armazenamento dos computadores. Os sistemas de hipertexto foram os precursores destes sistemas, e dedicam-se principalmente, ao armazenamento de informação textual. Os sistemas de hipertexto/hipermídia constituem uma experiência recente e bem sucedida para o armazenamento de documentos em meio eletrônico.

3.1. Conceituação

Um hipertexto caracteriza-se por apresentar a informação de forma não sequencial. Para transformar um texto tradicional em hipertexto, é preciso fragmentá-lo em trechos, e interligá-los através de caminhos de leitura. O leitor de um hipertexto liberta-se da forma sequencial de leitura, *navegando* pelos trechos do texto, através dos caminhos que os interligam. Os trechos e caminhos são conhecidos na taxonomia de hipertexto, respectivamente, como *nós* e *links*. Para passar de um nó para outro, os nós de origem oferecem acionadores de links que são conhecidos como *botões*. A figura 3.1 ilustra os conceitos de nó, link e botão. Nos sistemas de hipermídia, o conteúdo de um nó pode ser um texto, um gráfico, uma música, um filme ou até mesmo um código executável.

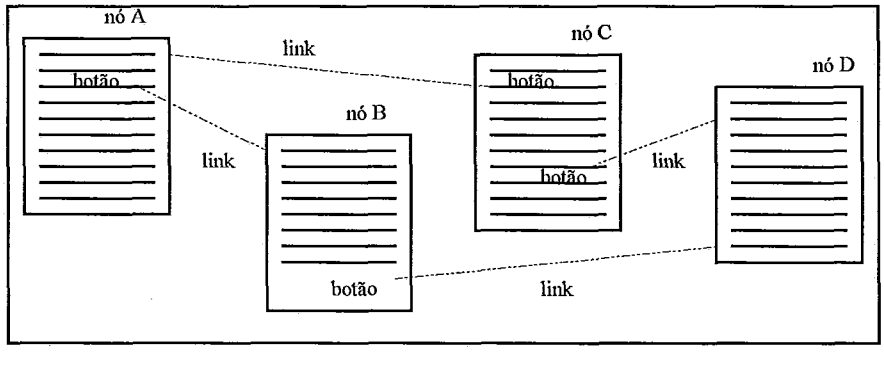


Figura 3.1: Nós, links e botões de um Hipertexto.

Os sistemas de hipertexto foram concebidos com objetivos específicos de armazenar e apresentar documentos na forma hipertextual: os *hiperdocumentos*. Em geral, estes sistemas oferecem facilidades de autoria e leitura de hiperdocumentos. Uma facilidade importante é a capacidade de atender a vários usuários simultaneamente, não só para leitura como também para a co-autoria de um hiperdocumento. Alguns destes sistemas possibilitam aos leitores uma participação ativa em relação ao hiperdocumento, permitindo a inclusão de *anotações*, como se faz nas margens dos documentos em papel. Desta forma, os leitores podem beneficiar-se dos comentários e esclarecimentos anotados por outros leitores. Por outro lado, os autores podem beneficiar-se dos observações, críticas e comentários feitos pelos leitores encarregados da revisão do hiperdocumento.

Outras facilidades interessantes são o *controle de versões* e os *marcadores*. O controle de versões facilita a co-autoria de hiperdocumentos, permitindo a coexistência de várias cópias de um mesmo hiperdocumento. Os marcadores permitem que o leitor marque e desmarque posições no hiperdocumento, como se faz durante a leitura de um livro.

A flexibilidade inerente ao formato não-linear dos sistemas de hipertexto, pode provocar a desorientação dos leitores, enquanto que em documentos lineares, como os livros, a localização é sempre imediata. Para superar esta dificuldade, os sistemas de hipertexto costumam oferecer mecanismos de *consulta*, *mapas* e *excursões*. Os mecanismos de *consulta* permitem que o leitor efetue a buscas por todo o hiperdocumento. Uma vez feita a busca de uma cadeia de caracteres, o leitor é transportado para os pontos onde foram encontradas ocorrências desta cadeia. Através destes mecanismos, o leitor pode navegar pelo hiperdocumento mais livremente, libertando-se da tirania dos links previamente criados.

Uma *excursão* é um conjunto de nós e links, que formam um caminho pré-definido através do hiperdocumento. As excursões são úteis para guiar o leitor novato. Alguns sistemas de hipertexto geram o resultado de consultas em forma de excursão, isto é, quando uma consulta tem como resultado mais de uma opção, o sistema gera dinamicamente uma excursão com os nós que satisfazem a esta consulta.

Os *mapas* podem ser globais, para visualização total da rede de nós do hiperdocumento, ou locais, para uma visualização parcial da mesma. Através dos mapas, os leitores podem orientar-se e movimentar-se pelos nós do hiperdocumento.

3.2. Aplicações

São diversas as áreas em que se pode aplicar o enfoque de hipertexto/hipermídia. Atualmente, pode-se encontrar aplicações interessantes, desenvolvidas para apoiar atividades que envolvem a organização e a difusão de informação. As áreas de marketing, educação e turismo por exemplo, utilizam sistemas de hipermídia para difundir a informação. Já as áreas de jornalismo, auditoria e pesquisa, apoiam-se em sistemas de hipertexto/hipermídia também para organizar a informação.

3.2.1. Difundindo a Informação

Sistemas de hipertexto genéricos têm sido usados para apresentar manuais de uso e manutenção de produtos (hipermanuais), tais como aparelhos eletrônicos, bicicletas, carros, navios e software [Niel93]. Uma das principais vantagens do uso de hipertextos, é permitir a configuração do manual conforme a experiência do usuário. Outra grande vantagem é a redução do volume e, consequentemente, dos gastos com a distribuição dos manuais. Alguns produtos são tão complicados que a distribuição de manuais impressos é praticamente inviável. Os hipermanuais ficam ainda mais interessantes, quando adicionam recursos multimídia, possibilitando aos usuários por exemplo, ver em animação o encaixe de uma peça do motor de um carro, ou ouvir a funcionalidade de cada botão do painel de um aparelho de som.

Os usuários de software dificilmente lêem os respectivos manuais. Na forma de hiperdocumento associado ao ambiente do próprio software, os manuais ficam ao alcance rápido e direto dos usuários. No momento em que se quer esclarecer uma dúvida ou aprender determinada facilidade do software, o usuário transporta-se para o hiperdocumento, obtendo diretamente, através de mecanismos de consulta, a informação desejada. Além deste acesso direto, os hipermanuais costumam oferecer excursões, que encaminham os usuários novatos através de tutoriais didáticos.

Publicações eletrônicas, como dicionários e enciclopédias, abrem novas possibilidades para a divulgação da informação. Os sistemas de hipermídia permitem a inclusão de ilustrações animadas e sonoras, o que torna o processo de transmissão da informação muito mais poderoso e eficiente. Enciclopédias como a Compton's, ilustram alguns assuntos com trechos de filme ou música. Já os dicionários incluem a sonorização da pronúncia correta de um vocábulo. Além destes recursos, a interconectividade dos hipertextos, facilita enormemente a busca e coleta da informação, abolindo o intenso folheio que ocorre normalmente em publicações impressas.

Os sistemas de hipertexto/hipermídia são também muito úteis para a criação de hiperdocumentos educativos. A interface amigável destes sistemas facilita aos educadores, a tarefa de autoria. Uma vez criado o hiperdocumento, os estudantes podem navegar pelas trilhas de informação, fazendo anotações sobre os textos e ilustrações, e ainda comunicando-se com outros estudantes e com o próprio professor. Os hiperdocumentos educativos estimulam o aprendizado, convidando os estudantes à exploração de idéias, temas e fatos por si próprios [Meyr86]. Entre hiperdocumentos educativos conhecidos, estão o Sistema Palenque, que ensina sobre a arqueologia Mexicana, o Projeto Encuentros, que ensina espanhol, o Projeto Shakespeare, que ensina dramaturgia [Niel93] e o Sistema América à vista [Roch92], que ensina sobre o encontro das culturas.

Para vender alguns produtos, o meio impresso é extremamente ineficiente. Na área musical e cinematográfica por exemplo, *clips* e *trailers* são veiculados através da televisão ou do cinema, para melhor atingir os consumidores. Atualmente, os catálogos hipermídia em CD-ROM's oferecem outra possibilidade para divulgação destes produtos. Mesmo para outros produtos, a utilização de catálogos hipermídia é interessante. Além de auxiliar o consumidor a escolher um produto, alguns sistemas permitem que em seguida, o consumidor solicite a compra do mesmo.

Uma estratégia de marketing interessante é observar o interesse dos consumidores para melhorar o enfoque da propaganda dos produtos. A monitoração do uso de catálogos hipermídia permite extrair dados interessantes. As estatísticas revelam por exemplo, que quando a relação procura/compra de alguns produtos está alta, isto pode significar que os consumidores que utilizam aquele catálogo, possuem baixo poder aquisitivo. A partir da análise destes estudos, os gerentes de marketing projetam novas versões de catálogos, mais adequados aos consumidores que querem incentivar.

Sistemas de hipertexto/hipermídia têm sido usados com muito sucesso para guiar visitantes e turistas. Locais como museus, bibliotecas, aeroportos, rodoviárias e shoppings de várias cidades do mundo, oferecem terminais com tela sensível ao toque, onde o visitante pode navegar por mapas e serviços oferecidos, localizando-se e planejando a sua visita.

3.2.2. Organizando a Informação

Auditores e jornalistas lidam diariamente com a coleta, organização e análise de documentos e informações provenientes de diversas fontes. Os sistemas de hipertexto/hipermídia podem ser úteis para aumentar a produtividade destas tarefas. Durante a coleta de documentos, pode-se trazer cópias de originais através da

utilização de "scanners", e armazená-los em nós de um hiperdocumento. A organização é facilitada na medida em que é possível criar links entre os documentos que têm alguma relação ou que fazem referência um ao outro. Por último, o auditor ou jornalista, desenvolve uma análise sobre a teia de documentos e informações recolhidas. Ao compor esta análise no ambiente do próprio hiperdocumento, ele finaliza seu trabalho de maneira integrada, criando botões dentro do texto da análise, que apontam para os nós contendo os documentos correspondentes. Uma vez terminada, a matéria jornalística ou a auditoria de uma empresa, o hiperdocumento gerado ainda pode ser usado como base para trabalhos posteriores.

A característica multi-usuário dos sistemas de hipertexto/hipermídia facilita ainda mais o trabalho destes profissionais, quando existe uma equipe trabalhando em um mesmo assunto ou empresa. Outra vantagem da utilização destes sistemas, é a possibilidade de acompanhamento do trabalho em realização. O gerente de uma equipe de auditores, por exemplo, pode obter informações do tipo: quais documentos foram coletados sobre determinado departamento da empresa, o que foi analisado por determinado membro da equipe, e assim por diante.

O trabalho de pesquisa é análogo ao trabalho de auditores e jornalistas. Ao iniciar uma pesquisa, o pesquisador seleciona publicações acadêmico-científicas de seu interesse. À medida em que reúne estas publicações, o pesquisador precisa organizá-las e rotulá-las, para mais tarde referenciá-las em um texto de sua autoria. Com o crescente número de publicações em meio eletrônico, a tarefa de coleta e organização tende a tornar-se cada vez mais fácil e rápida.

Assim como para a organização de documentos, os sistemas de hipertexto/hipermídia são úteis para a organização de idéias. No processo de criação, seja individual ou em grupo, é difícil coordenar a geração de idéias. Os sistemas de hipertexto, permitem a captura e organização destas idéias em um único hiperdocumento. Este hiperdocumento é ao mesmo tempo, organizador e fomentador de idéias, pois ao vizualizar tudo o que foi cogitado até o momento, o usuário pode conceber novas e melhores idéias.

O trabalho do engenheiro de software também envolve a organização de idéias e documentos. A próxima seção é especialmente dedicada a esta aplicação de sistemas de hipertextos.

3.3. Hipertextos e a Engenharia de Software

A Engenharia de software é uma das áreas que mais se beneficia com a utilização de sistemas de hipertexto. Seu produto final, o código fonte, reside obrigatoriamente em meio eletrônico, e está vinculado a outros tipos de documentos, gerados durante as fases de seu desenvolvimento. A reunião de todos estes documentos em um único ambiente, permite que os documentos referenciem-se entre si, facilitando o entendimento do sistema.

Mesmo que um documento gerado durante o processo de desenvolvimento de software, não esteja vinculado aos demais produtos deste processo, a sua apresentação sob a forma de hipertexto é vantajosa. Considerando que um hiperdocumento sobre o sistema, é mais fácil de compreender do que a sua versão sequencial, os custos com a manutenção e suporte deste sistema serão minimizados. Os hipermanuais de software, citados na seção anterior, são exemplos em que a apresentação hipertextual de um documento de software tem mostrado uma grande aceitação.

O ambiente Dynamic Design desenvolvido na Tektronix [BiRi87, Bige88] armazena todos os documentos gerados durante o desenvolvimento de um sistema (artefatos de software) escrito em C, sob a forma de hipertexto. Cada documento é armazenado em um nó e pode interconectar-se com outros documentos. Os documentos são agrupados segundo o contexto a que pertencem, conforme mostra a figura 3.2. As linhas que ligam dois contextos, representam que há links entre documentos destes contextos.

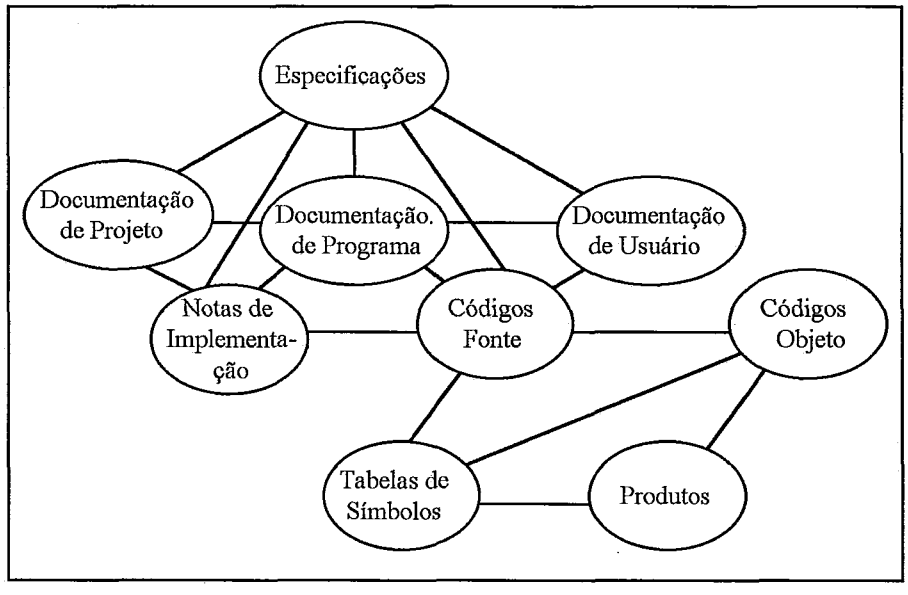


Figura 3.2: Contextos dos artefatos de software no ambiente Dynamic Design [Bige88].

O ambiente DIF ("Documents Integration Facility"), criado por Garg e Scacchi [GaSc90], é um sistema baseado em hipertexto semelhante ao Dynamic Design. O DIF

foi projetado para apoiar a documentação das várias versões do software pelas diversas fases do seu processo de desenvolvimento [Borg93]. Assim como o Dynamic Design, o DIF armazena especificação de requisitos, especificações funcionais, projeto arquitetônico e detalhado, código fonte, informação de teste, e manuais de usuário e de manutenção, em nós de um hipertexto. A principal diferença entre estes dois ambientes é que o Dynamic Design gerencia os documentos referentes a um único sistema, enquanto o DIF pode controlar a documentação de diversos sistemas.

O sistema HyperMan desenvolvido pela Technical University of Munich [SaMyGü92] possui um enfoque mais voltado para a Engenharia Reversa. Este sistema converte toda a documentação de um software, código fonte e outros documentos (textos), para um único hiperdocumento. Seu objetivo é explicitar as relações entre os programas e os trechos de textos correspondentes, assim como interrelacioná-los dentro de si mesmos, de modo que estas ligações facilitem o entendimento e a manutenção do software. O hiperdocumento é usado apenas para leitura, obrigando que qualquer alteração seja feita nos documentos originais. No entanto, como a conversão é feita de forma totalmente automática, a atualização do hiperdocumento em relação aos documentos alterados é fácil e imediata.

Existem sistemas de hipertexto menos abrangentes, que foram projetados para apoiar apenas a fase de Projeto de Software. O gIBIS ("graphical Issue-Based Information System") da MCC ("Microelectronics and Computer Technolgy Corp.), é um sistema de hipertexto multi-usuário, projetado para capturar o raciocínio que leva às decisões de Projeto de um sistema [CoBe87, Niel93]. Diferente dos sistemas de hipertexto anteriormente citados, esta ferramenta suporta o trabalho cooperativo, permitindo que algumas pessoas interajam entre si durante o projeto de um sistema. O gIBIS baseia-se em um modelo de argumentação chamado IBIS [KuRi70], que possibilita que os participantes tragam seu conhecimento para resolver questões sobre o projeto. Para cada questão, os participantes sugerem posições e argumentam contra e a favor destas. Toda a discussão é registrada em um hiperdocumento com nós do tipo *questão, posição* e *argumento*, e links do tipo *suporta, objeta, responde, questiona, sugere, substitui, generaliza e especializa* (veja a figura 3.3).

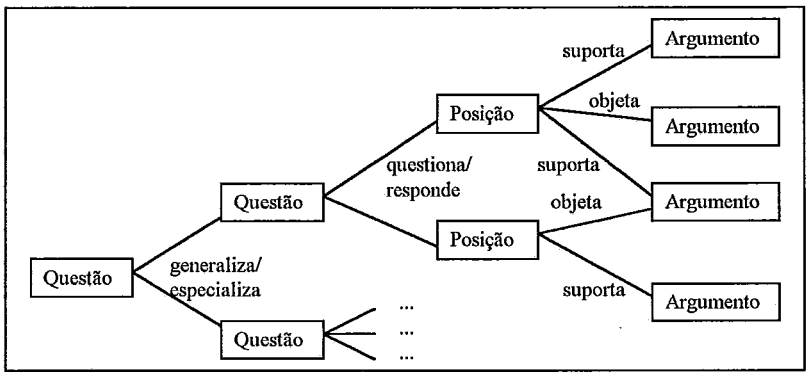


Figura 3.3: Esquema do hiperdocumento de uma dicussão usando o modelo IBIS.

A característica gráfica do gIBIS provém do fato de que ele oferece os mapas local e global do hiperdocumento. Ao mesmo tempo em que o usuário visualiza o conteúdo de um nó, ele pode localizar-se em relação à toda a teia da discussão, podendo inclusive, transportar-se para qualquer outro ponto da mesma.

Os sistemas de hipertexto suprem a necessidade de armazenamento e visualização da informação contida nos documentos de software de forma bastante satisfatória. Na Engenharia Reversa, a recuperação da informação sobre um sistema, requer a escolha de uma representação para a mesma, sendo a forma hipertextual uma opção bastante interessante. No entanto, o processo de recuperação desta informação, muitas vezes requer a participação e o envolvimento de um grupo de pessoas. Embora existam sistemas de hipertexto que são multi-usuário, para que estes sistemas apoiem adequadamente a interação entre várias pessoas, é preciso contar com tecnologias especialmente dedicadas ao suporte do Trabalho Cooperativo.

Capítulo 4
Trabalho Cooperativo Suportado por Computador

Afirmações como "estudos estatísticos mostram que 30% a 70% do tempo de um trabalhador de escritório é gasto em reuniões" [Stef87], ou "a maior parte do nosso empenho intelectual envolve a colaboração de outras pessoas" [Land93], mostram que por trás da produção de um trabalho, mesmo que de autoria individual, há quase sempre o envolvimento de mais de uma pessoa. Esta constatação tem motivado muitas pesquisas para o desenvolvimento de tecnologias de suporte ao trabalho cooperativo, seja para comunicação entre os membros do grupo, como para conduzir o trabalho em si. As pesquisas que apostam na utilização de computadores pertencem à área conhecida como Trabalho Cooperativo Suportado por Computador, ou CSCW ("Computer Supported Cooperative Work").

4.1. Histórico

Em meados dos anos 70, o CSCW apareceu sob o nome de Automação de Escritório (OA - "Office Automation"), a partir da necessidade de transformar aplicações mono-usuário, como processadores de texto e planilhas eletrônicas, em aplicações multi-usuário [Grud94]. Só em meados dos anos 80, quando reconheceu-se a necessidade de aprender sobre o comportamento de um grupo em uma certa atividade, surgiu a sigla CSCW. Nesta época, os tecnólogos aliaram-se aos profissionais de áreas humanas, como por exemplo, sociólogos, psicólogos, antropólogos e educadores, buscando o desenvolvimento de tecnologias mais adequadas ao suporte do trabalho cooperativo.

O termo "groupware" costuma ser usado quase como sinônimo de CSCW, porém alguns autores identificam uma tendência diferenciada no emprego destes termos. Enquanto CSCW é usado para designar a pesquisa na área do trabalho em grupo e como os computadores podem apoiá-lo, groupware tem sido usado para designar os produtos comerciais desta área [Grud94, ElGiRe91].

A maior parte da pesquisa sócio-psicológica disponível, enfoca uma observação pouco detalhista do comportamento de grupos. Além disso, em geral, os grupos estudados são aleatoriamente formados por estudantes [Olso93]. Para projetar ferramentas adequadas, os tecnólogos precisam de melhor embasamento teórico, isto é, precisa-se conhecer os detalhes do comportamento de grupos em situações reais de trabalho. A

etnologia é um ramo da antropologia que estuda a cultura dos povos, e que pode fornecer esta base teórica, através da realização de estudos etnográficos.

A etnografia é uma disciplina da etnologia, cujo objetivo é o estudo e a descrição detalhada do comportamento de grupos de pessoas [Blom93]. Em seu trabalho, os etnógrafos costumam ir a campo e submergir nas atividades das pessoas em estudo, valendo-se de técnicas de observação, entrevistas e análise de vídeo, para auxiliar no entendimento e descrição destas atividades. Os resultados de estudos etnográficos são fundamentais na construção de uma teoria mais rica para apoiar a pesquisa em CSCW.

Ao invés de esperar pela teoria, a comunidade interessada em CSCW passou a construir groupwares, sobre os quais desenvolvem-se estudos etnográficos. Embora esta precipitação dê origem a produtos pouco adequados, as descobertas feitas a partir de seu uso têm fornecido subsídios para a construção de groupwares melhores.

4.2. Classificação

Estão começando a surgir tentativas de classificação dos groupwares, num esforço de entendimento de como a tecnologia pode se encaixar no contexto social, organizacional e cultural do homem. Olson et al. [Olso93] descrevem quatro grandes tipos de groupware: sistemas de suporte a reuniões, sistemas de conferência e correspondência eletrônica, sistemas de co-autoria e sistemas de vídeo-conexão.

4.2.1. Sistemas de Suporte a Reuniões

Os sistemas de suporte a reuniões foram desenvolvidos para apoiar diferentes tipos de trabalho em grupo face-a-face. Dois dos principais tipos de sistemas que se encaixam neste grupo são as salas de reunião eletrônicas (EMS - "Eletronic Meeting Systems") e os sistemas de suporte à tomada de decisão em grupo (GDSS - "Group Decision Support Systems"). Os EMS's oferecem ambientes especiais com grande suporte de hardware e software, para apoiar reuniões face-a-face [Borg93]. A Universidade do Arizona possui um dos mais conhecidos sistemas deste tipo, o PlexCenter Planning and Decision Support Laboratory, cujo ambiente compõe-se de uma mesa de conferência em forma de U com estações de trabalho interligadas entre si e conectadas a um telão. Este telão pode mostrar uma das telas da mesa ou a compilação de algumas delas, conforme o comando do líder da reunião, também conhecido como "facilitador" [Nuna91,ElGiRe91].

Os GDSS's têm o objetivo de aumentar a produtividade de reuniões decisórias e melhorar a qualidade de seu resultado, através da estruturação e captura do raciocínio e argumentação gerados durante a reunião. Em geral os GDSS's baseiam-se em modelos de argumentação. Entre os modelos existentes podemos citar:

- **IBIS**, usado pelas ferramentas gIBIS e rIBIS [YaCo90, CoBe87, ReEl91];
- **QOC**, usado pelo Design Space Analysis (DSA) [ShHa94];
- **DRL**, usado pela ferramenta SIBYL [Lee90];

O modelo IBIS - "Issue-Based Information System" - compreende três elementos: questão[1], posição e argumento, que correspondem respectivamente, aos problemas em discussão, às possíveis soluções para os problemas, e às opiniões favoráveis ou não às soluções levantadas. Uma questão pode ter várias posições para resolvê-la, e cada posição de uma questão pode, por sua vez, ter um ou mais argumentos favoráveis ou não. A figura 4.1 ilustra os elementos do modelo IBIS e seus relacionamentos.

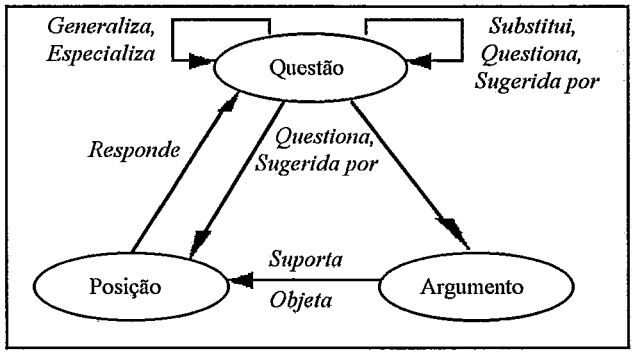


Figura 4.1: Modelo IBIS

Semelhante ao modelo IBIS, o modelo QOC, "Question, Option, Criteria", envolve quatro elementos: questão, opção, critério e argumento. As *questões* são os problemas chave a resolver, as *opções* são as alternativas levantadas para resolver os problemas identificados, os *critérios* justificam as opções existentes, e finalmente, os *argumentos* são usados para conduzir a discussão sobre os demais elementos (veja a figura 4.2).

[1]Tradução do termo em inglês "issue".

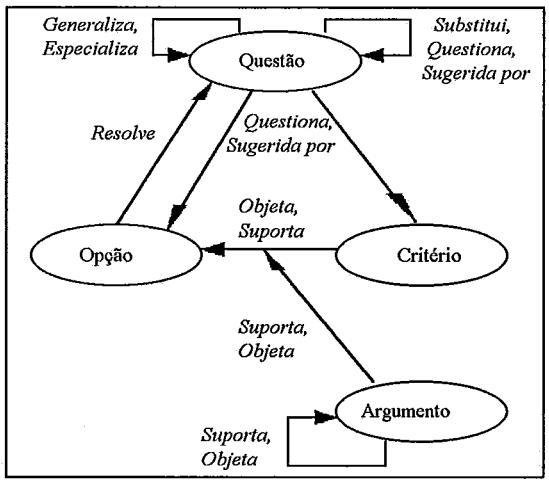


Figura 4.2: Modelo QOC

O modelo DRL, "Design Rationale Language", é considerado uma extensão da notação IBIS [Lee90], pois inclui, explicitamente, a noção de *"objetivos"*. Os *objetivos* estão relacionados ao *problema* em discussão, sendo as *alternativas* de solução avaliadas através de *alegações*, em relação a estes objetivos. No modelo IBIS os objetivos estão implícitos nos argumentos que apoiam as alternativas (posições). A figura 4.3 mostra o modelo DRL.

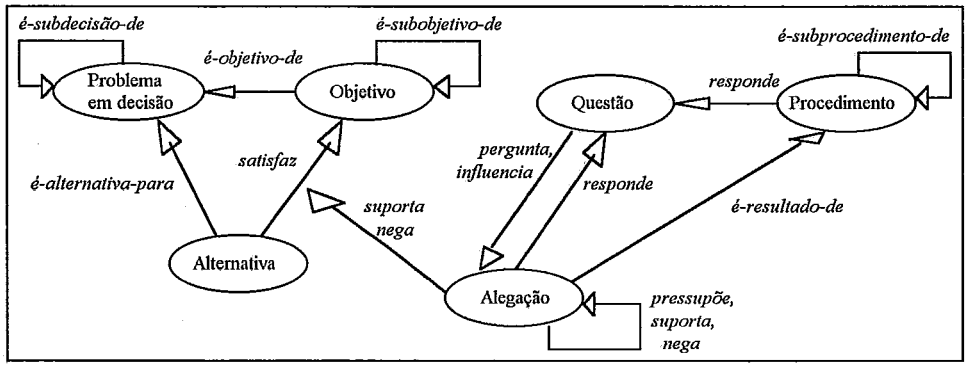


Figura 4.3: Modelo DRL

Nem todas as reuniões são decisórias. Alguns grupos se reúnem para discutir um assunto, avaliar um documento, ou simplesmente estudar. Os sistemas que apoiam este tipo de reunião são conhecidos como Sistemas de Apoio à Discussão ou Organizadores de Idéias [Stef87]. O modelo DRL é mais complexo e voltado para a tomada de decisões, enquanto os modelos IBIS e QOC, são mais simples e genéricos, podendo ser usados para apoiar discussões.

4.2.2. Sistemas de Conferência e Correspondência Eletrônica

Os sistemas de conferência e correspondência eletrônica (correio eletrônico) são aqueles que apoiam a comunicação a longa distância, síncrona (simultânea) e

assíncrona, respectivamente. Em geral, estes sistemas permitem a interação através da palavra escrita, mas a ausência de entonação pode provocar uma má interpretação do que se quer dizer. Este problema tem sido contornado pela crescente utilização de símbolos formados por conjuntos de caracteres, com significados padronizados, que quando aparecem ao lado de uma frase informam o tom da mesma.

Os sistemas de conferência eletrônica permitem que grupos de usuários interajam sincronamente. Muitos destes sistemas são simples editores de textos cuja saída (texto em edição) pode ser vista por quantos usuários estiverem interagindo, e cuja entrada (poder de edição) é permitida a um dos usuários por vez. Os sistemas de correio eletrônico são gerenciadores de mensagens. A proliferação deste tipo de sistema acarretou no que se chama de "sobrecarga de mensagens" [ElGiRe91]. São tantas as mensagens recebidas, que o usuário não consegue ter tempo para processá-las. Algumas aplicações já incorporam certa "inteligência", permitindo a classificação automática das mensagens de acordo com seu conteúdo. Um exemplo de aplicação inteligente é o Information Lens [Olso93, ElGiRe91], onde o usuário pode especificar regras para arquivar ou redirecionar mensagens automaticamente.

4.2.3. Sistemas de Co-autoria

Os sistemas de co-autoria são editores multi-usuários. Basicamente, há dois tipos de sistemas de co-autoria: síncronos (em tempo real) e assíncronos. Os sistemas síncronos permitem que um grupo de pessoas editem o mesmo documento ao mesmo tempo. Nestes sistemas o documento compartilhado é dividido em segmentos. Cada segmento pode ser visualizado por todos os autores, mas só um autor de cada vez pode ser editar um segmento. Nos sistemas assíncronos, há um autor por vez e vários revisores. Os revisores têm acesso a uma área separada, onde podem fazer comentários sobre o texto ou anotar idéias. PREP, QUILT, ForComment e FolioViews são aplicações para a co-autoria assíncrona, enquanto GROVE e ShrEdit são aplicações para a co-autoria síncrona [Borg93, Olso93, Olso92].

Landow identifica quatro modos de co-autoria [Land93]: lado a lado, versão, linha de montagem e hipertexto. O modo lado a lado implica em que os autores compõem ao mesmo tempo. Mais comumente usado, o modo versão envolve a composição de esboços, que ora estão sob a autoria de um, ora sob a autoria de outro co-autor. No modo linha de montagem, a composição é dividida em várias sessões, e estas são distribuídas entre os co-autores, que trabalham independentemente. Finalmente, o quarto modo é uma composição dos modos versão e linha de montagem. Como foi

visto anteriormente, em um hipertexto o texto está dividido em vários trechos, como ocorre no modo linha de montagem. Os sistemas de hipertexto incorporam também o modo de versão, pois possibilitam a convivência simultânea de versões individuais (privadas) e uma versão comum a todos os autores (pública). Ferramentas como a Intermedia [Land93] e o FolioViews [Foli93] por exemplo, implementam o quarto modo de co-autoria.

A coordenação de um grupo de co-autores é um dos problemas que os sistemas de co-autoria precisam contornar. Uma resposta para este problema é a definição de *papéis* sociais, aos quais atribui-se responsabilidades e um padrão de interação com os demais [Neuw90]. A ferramenta QUILT por exemplo, define três papéis: o co-autor, o comentador e o leitor, e a cada papel é atribuído um conjunto de ações diferente. A definição de papéis, no entanto, pode se tornar restritiva ou até mesmo prejudicial, se defnida inadequadamente. Para tirar um bom proveito desta estratégia, é necessário pesquisar bastante os aspectos sociais do ambiente de co-autoria em questão.

Uma aplicação menos comum dos sistemas de co-autoria assíncrona, é o suporte a grupos que trabalham por turno, como geralmente ocorre em hospitais e fábricas. Há grupos que mesmo trabalhando no mesmo local e no mesmo turno quase não se encontram. Muitas vezes estes grupos desempenham atividades secundárias que precisam conviver com a falta de sincronismo de seus membros, que por motivos de atividades prioritárias ou mesmo por falta de organização, não conseguem encontrar tempo para desempenhá-las. Neste caso, os sistemas de co-autoria assíncrona também podem ser bastante úteis.

4.2.4. Sistemas de Vídeo-Conexão

Os sistemas de vídeo-conexão são aqueles que permitem reuniões distribuídas em que participantes remotamente dispostos podem "ver" uns aos outros, bem como gráficos, diagramas ou algum outro documento de que trata a reunião. CAVECAT e ClearBoard são dois exemplos de sistema de vídeo-conexão [Olso93]. Enquanto o primeiro mostra os participantes em pequenas janelas na tela, o segundo dispõe os usuários ao redor do objeto em discussão, com se olhassem através de um vidro. Outro tipo de sistemas de vídeo-conexão é aquele que permite a visualização de escritórios com o objetivo de verificar a disponibilidade destes ou das pessoas que lá se encontram trabalhando. O CRUISER é um exemplo de sistema deste tipo [ElGiRe91].

4.2.5. Classificação Espaço/Tempo

A classificação espaço/tempo de DeSanctis e Gallupe apresentada em 1987 [DeGa87], tem sido largamente utilizada [Grud94] e também divide as aplicações groupware em quatro categorias. A figura 1 mostra como as aplicações comentadas até então, encaixam-se na classificação espaço/tempo.

	ao mesmo tempo	**em momentos diferentes**
no mesmo lugar	GDSS's, EMS's, Sistemas de Apoio à Discussão	co-autores assíncronos para apoio ao trabalho por turnos.
em diferentes lugares	conferência eletrônica, co-autores síncronos, vídeo-conexão	correio eletrônico, co-autores assíncronos

Figura 1: Classificação Espaço/Tempo

4.3. CSCW e a Engenharia de Software

"O trabalho da equipe responsável pelo desenvolvimento de software é essencialmente cooperativo, mesmo considerando que os membros do grupo mudam durante as diversas fases do processo." [Borg93]

As cinco fases do ciclo de vida clássico de um software denominam-se: análise, projeto, implementação, verificação e manutenção. Em cada uma destas fases, são desempenhadas atividades em grupo, cujo produto é conhecido como "artefato de software". A utilização de tecnologias de CSCW contribui bastante para melhorar a produtividade deste grupo e a qualidade dos artefatos de software resultantes. A tabela 4.1 mostra a correspondência entre os tipos de groupware e as fases do ciclo de vida do software que podem apoiar.

Ciclo de vida	**Groupware**
análise	sistemas de suporte a reuniões
projeto	sistemas de suporte a reuniões
implementação	sistemas de co-autoria, sistemas de conferência eletrônica
verificação	sistemas de suporte a reuniões, sistemas de co-autoria, sistemas de conferência eletrônica
manutenção	sistemas de suporte a reuniões, sistemas de conferência eletrônica

Tabela 4.1: Groupware para apoiar o ciclo de vida do software.

Nas fases de análise e de projeto, além do artefato resultante, o raciocínio e as argumentações que levaram a produção do artefato, são informações importantes [PoBr88, PrLuLe91]. A captura da argumentação nas fases de análise ou projeto de um software pode trazer diversos benefícios, tais como a organização dos assuntos a discutir, a manutenção da coerência e consistência da discussão através das várias reuniões realizadas, a facilidade de integrar novos membros à discussão em andamento, e finalmente, a possibilidade de reutilizar aquela argumentação nas fases de análise ou projeto de um outro software.

Na fase de análise a participação do usuário é fundamental. Muitas vezes a distância física entre usuários e engenheiros de software impõe dificuldades para a realização de reuniões com a frequência necessária. Grudin em [Grud91b] sugere a utilização de sistemas de vídeo-conexão para habilitar uma comunicação mais efetiva entre usuários e desenvolvedores, aproximando as suas diferentes realidades e ambientes.

Os sistemas de co-autoria são ótimas ferramentas para apoiar as tarefas da fase de implementação. Em um ambiente de programação distribuído faz-se necessário um controle de versões dos programas fonte. O armazenamento centralizado destes não é suficiente, pois dois programadores podem estar alterando o mesmo programa ao mesmo tempo. Neste caso, o último a terminar as alterações terá sua versão do programa sobreposta à versão do outro programador. Os sistemas de co-autoria encarregam-se deste controle, permitindo no modo assíncrono, que apenas um programador edite um arquivo fonte, enquanto os demais podem apenas visualizá-lo e comentá-lo. Quando é preciso que dois programadores editem diferentes partes do mesmo fonte, basta ativar o modo síncrono dos sistemas de co-autoria que oferecem esta facilidade.

A depuração de programas é uma tarefa que ocorre durante as fases de implementação, testes e manutenção. Em geral, os programadores perdem muito tempo nesta tarefa, custando a encontrar onde está o erro do programa. Muitas vezes o programador busca a ajuda de um colega, para que juntos consigam resolver a questão. Para apoiar a depuração cooperativa entram em cena os sistemas de conferência eletrônica, que permitem que dois ou mais programadores, cada qual em sua estação de trabalho, acompanhem a execução do programa em verificação, e troquem comentários e observações durante a mesma.

A Inspeção é um método muito usado durante a fase de verificação, para avaliar o código fonte de um programa sem submetê-lo a execução experimental. Além do

programa, as Inspeções também são utilizadas para a verificação de produtos de outras fases, como por exemplo os diagramas de estrutura, fluxo de dados e entidade-relacionamento [Your89b].

A realização de uma Inspeção conta com um grupo de pessoas do qual participam o autor do programa e um moderador (líder) para conduzir as reuniões do grupo, bem como alguns avaliadores [Roch87, Somm89, Your89b]. Antes das reuniões do grupo, os avaliadores devem se preparar para as mesmas através da revisão individual do código, em busca de falhas e pontos obscuros. Os sistemas de co-autoria assíncronos podem ser bastante úteis para esta atividade, na medida em que permitem que os avaliadores revisem, comentem e anotem sobre o mesmo documento, o código em inspeção. As reuniões de inspeção podem utilizar um sistema de apoio a discussão ou decisão, que se encarregará de estruturá-las e capturar o que for discutido durante as mesmas. O documento resultante servirá para a posterior referência do autor do programa, responsável pela efetuação das mudanças solicitadas.

Assim como a Engenharia Progressiva, a Engenharia Reversa também necessita do apoio de tecnologias CSCW. Atividades como a Redocumentação e a Recuperação de Projeto também mobilizam grupos de pessoas, o que as torna usuárias de aplicações groupware. Em especial, a Recuperação de Projeto necessita de apoio tanto para conduzir o trabalho de extração de informação, como para gerar o Projeto Arquitetônico de um sistema. No entanto, até onde se sabe, as ferramentas existentes para apoiar estas atividades não suportam o trabalho cooperativo.

Capítulo 5
O Ambiente para a Recuperação do Projeto Arquitetônico de Sistemas

Neste trabalho optamos pelo investimento no processo reverso, acreditando ser uma abordagem mais direta, para atacar os problemas da manutenção de software. Mais especificamente, focalizamos a atividade de Recuperação de Projeto de software, esperando aumentar a sua manutenibilidade, reduzir os custos com a sua compreensão e evitar a descaracterização e degradação do mesmo.

Propomos então, um ambiente adequado à tarefa de Recuperação da documentação relativa ao Projeto Arquitetônico de sistemas. O enfoque adotado por esta proposta, visa recuperar esta documentação a partir do código fonte e do conhecimento dos especialistas no software. No ambiente proposto a tarefa de Recuperação, divide-se em duas etapas: automática e cooperativa. Em ambas as etapas conta-se com o apoio de ferramentas automáticas. Para a primeira etapa é necessária a utilização de um analisador de códigos fonte. A segunda etapa apóia-se em ferramentas de hipertexto e trabalho cooperativo.

São dois os documentos recuperados: o Projeto Arquitetônico propriamente dito, e um documento contendo as informações trocadas entre os especialistas. Para representar o Projeto Arquitetônico utilizamos a notação HTDN ("HyperText Design Notation"), enquanto que para capturar o conhecimento dos especialistas utilizamos o modelo de argumentação IBIS ("Issue-Based Information System"). A figura 5.1 mostra um esquema para visualizar o ambiente proposto.

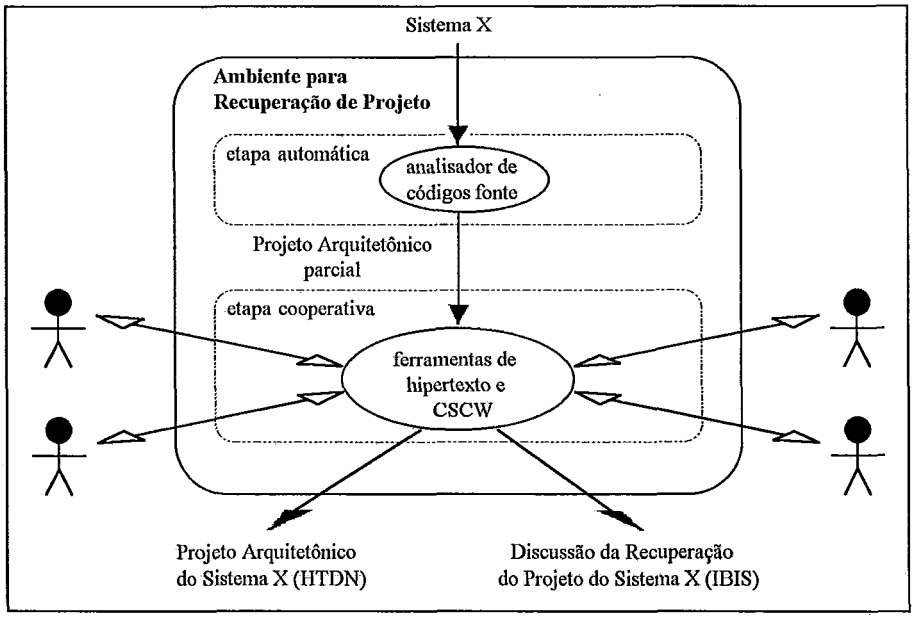


Figura 5.1: Esquema do Ambiente de Recuperação do Projeto Arquitetônico

5.1. Representando o Projeto Arquitetônico

A fase de Projeto pode ser dividida em duas subfases [Ghez91]: o Projeto Arquitetônico ou Preliminar, e o Projeto Detalhado (veja a figura 5.2). O Projeto Arquitetônico gera o que se conhece como Estrutura Modular ou Arquitetura do Sistema. A segunda fase consiste em um processo de refinamento da estrutura gerada pela primeira fase, gerando uma especificação detalhada de cada módulo, em um formato adequado para a fase de implementação.

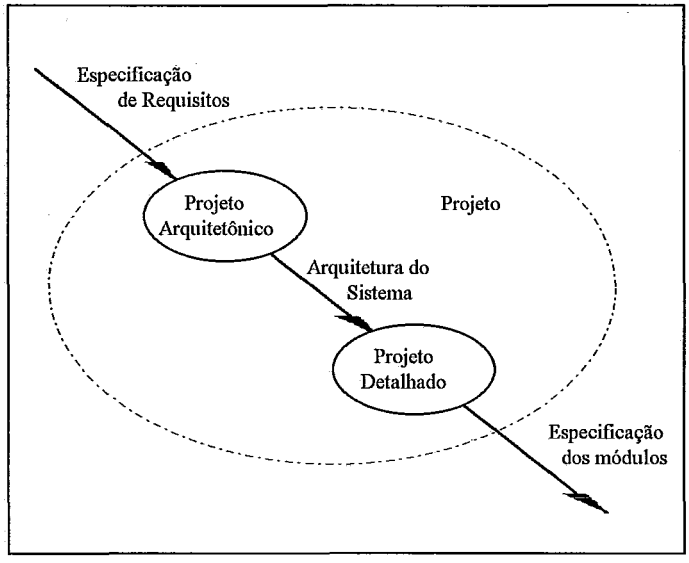


Figura 5.2: A fase de Projeto.

Na Engenharia Progressiva, o Projeto Arquitetônico é obtido através de um processo de modularização. Este processo envolve a identificação dos módulos que compõem a estrutura do sistema e a identificação dos relacionamentos existentes entre eles. Também na Engenharia Reversa estas duas atividades são necessárias, diferindo apenas na fonte de informação. Enquanto na Engenharia Progressiva parte-se da especificação do sistema, no processo reverso utiliza-se os documentos de menor abstração existentes, como por exemplo o código fonte. Entretanto, independente da forma de obtenção do Projeto Arquitetônico de um sistema, é preciso delimitar o que se considera uma documentação que o represente.

5.1.1. Notações Existentes

Entre as notações para representar o projeto de um sistema, existem as notações gráficas e textuais. O gráfico da Estrutura Modular ou Diagrama de Estrutura, desenvolvido para apoiar o método de Projeto Estruturado, é uma das mais utilizadas (veja a figura 5.3). Mas, embora a notação gráfica permita uma visualização imediata, facilitando a compreensão do sistema, ela é incompleta pois, por motivos de

simplicidade, omite informações, tais como detalhes sobre a interface entre módulos. Já a notação não gráfica, embora de difícil visualização, permite uma descrição completa do Projeto.

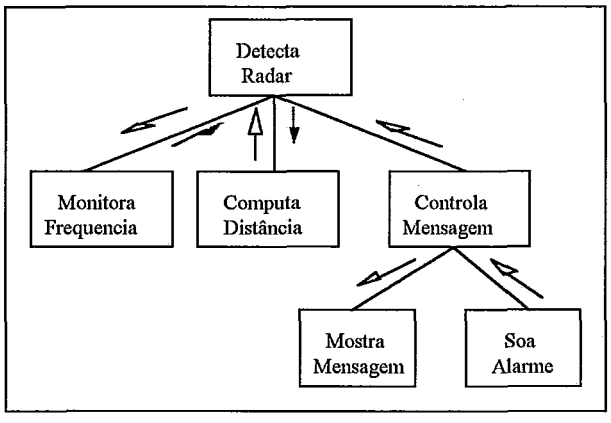


Figura 5.3: Diagrama de Estruturas.

Ghezzi et al. em [Ghez91], apresentam uma notação textual chamada "Textual Design Notation" (TDN). A notação TDN comporta informações formais e informais (veja a figura 5.4). A informação formal pode ser obtida a partir de uma análise do sistema em si, enquanto que a informação informal, como os comentários e a descrição do módulo, provém do conhecimento dos especialistas.

A TDN descreve a Estrutura Modular do sistema, ou seja, ela identifica e descreve cada módulo do sistema, e provê a relação *usa* entre eles. O conjunto de descrições de todos os módulos completa a documentação do sistema. Entretanto, se o número de módulos for razoavelmente grande, apenas a representação da relação *usa* é insuficiente. Além desta relação, a TDN provê também a relação *é-composto-por*. Nesta relação, os módulos do sistema são agrupados em outros módulos, conhecidos como *clusters*. O processo de *clusterização* tem o objetivo de criar *clusters* que representem um conjunto de outros módulos em termos mais abstratos, isto é, que permita a visualização de um conjunto de módulos através de um único conceito. Ao final deste processo a relação *é-composto-por* resulta em uma estrutura hierárquica conhecida como *árvore conceitual*. A figura 5.5 ilustra graficamente as duas relações discutidas.

Módulo Detecta-Radar
Usa Monitora-Frequência, Computa-Distância,
Controla-msg
Interface
var Cod-radar: integer;
type freq-comum: array (1..10) of real;
<comentários sobre a interface>
Implementação
<descrição do módulo>
É-composto-por Inicializa, Transf-Freq
fim Detecta-Radar

Figura 5.4: Notação TDN

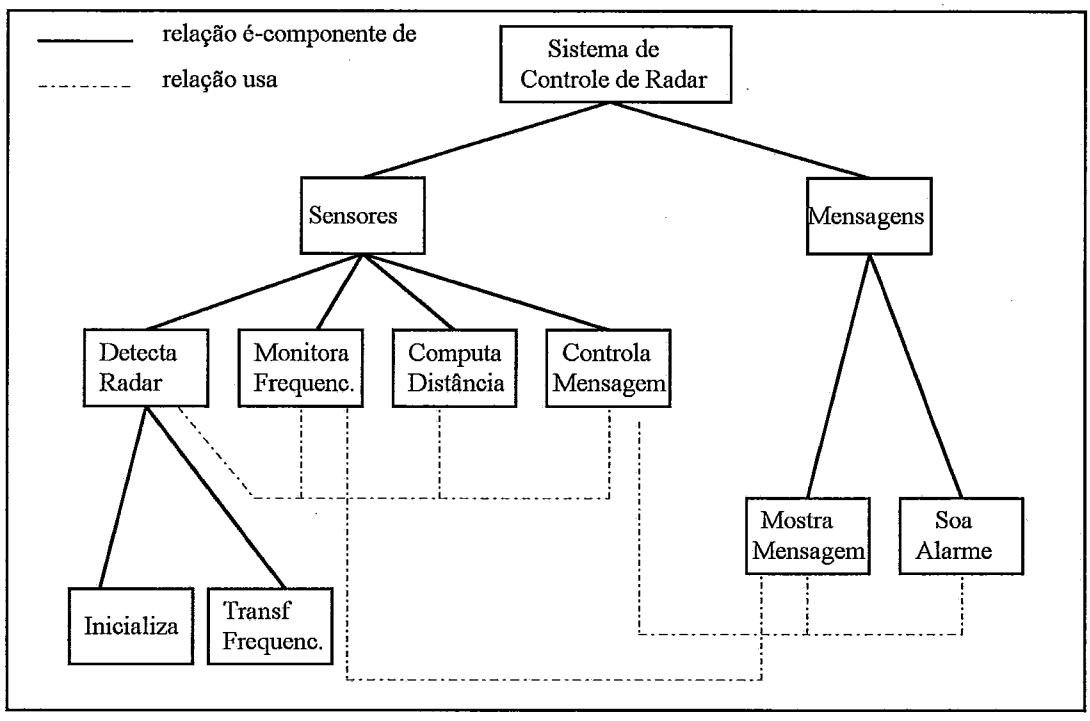


Figura 5.5: Árvore Conceitual.

Segundo Schwanke e Platoff [ScPl89] a maioria dos sistemas existentes já apresentam alguma *clusterização*. No entanto, esta *clusterização* pode não ser ideal. Existem alguns trabalhos que apresentam critérios e algorítmos para a geração da árvore conceitual. Os autores Choi e Scacchi [ChSc90], e Selby e Basili [SeBa91] apresentam uma *clusterização* baseada no acoplamento entre os módulos. Estes dois trabalhos objetivam o controle da propagação de erros (efeitos colaterais) ocasionada por alterações no sistema. Já o trabalho de Schwanke e Platoff [ScPl89] busca uma representação semântica do sistema. Através da análise das relações de uso dos módulos, considerando os dois sentidos da relação *usa*, extrai-se uma medida de similaridade entre os pares de módulos. Esta medida é usada para construir a árvore conceitual do sistema.

Independente da forma como se pretende extrair a ávore conceitual, é importante que a notação utilizada para o Projeto Arquitetônico permita a representação tanto da relação *usa,* como da relação *é-composto-por*. A TDN satisfaz esta condição, representando ambas as relações simultaneamente, porém sua forma textual dificulta a visualização destas relações.

Notações textuais e gráficas como as discutidas acima, destinam-se à apresentação do documento de Projeto Arquitetônico em papel. Embora o papel seja um meio barato e portátil, alguns autores, como Theodor Nelson e Douglas Engelbart, defendem que o meio eletrônico oferece mais recursos para a apresentação e o armazenamento de documentos [YaMeDa85]. No sistema Rigi [Muti92] por exemplo, utiliza-se o meio

eletrônico para apresentar e manipular o Projeto Arquitetônico do Sistema, em uma notação gráfica tridimensional. Outro exemplo da utilização dos recursos do meio eletrônico é o enfoque de hipertexto, que tem sido abordado em diversas áreas para melhorar a visualização de informações.

O Projeto Arquitetônico é um documento, cujo armazenamento e apresentação sob a forma de hipertexto é vantajoso, beneficiando-se especialmente do poder de conectividade deste enfoque. Em primeiro lugar, esta característica permite vinculá-lo diretamente ao código implementado. Em um hipertexto, é possível criar links que liguem cada módulo de um sistema descrito no documento de Projeto, ao arquivo com o código fonte correspondente.

Em segundo lugar, as relações *usa* e *é-composto-por* podem ser facilmente representadas em um hipertexto. Basta fragmentar o documento de modo a considerar cada módulo como um nó, e criar links interligando-os de acordo com os relacionamentos existentes. Uma vez criada a teia de módulos, o usuário pode navegar pela estrutura modular e/ou pela árvore conceitual do sistema. Caso o documento original do Projeto Arquitetônico utilize uma notação textual como a TDN, a sua forma hipertextual facilita a visualização do mesmo (veja a figura 5.6).

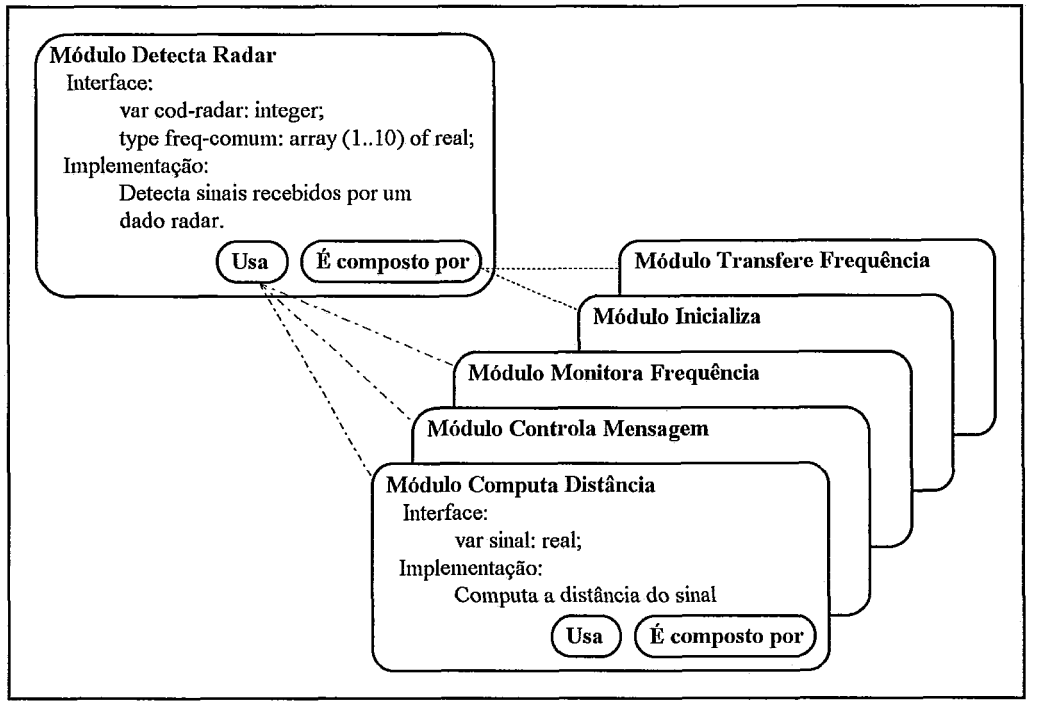


Figura 5.6: Notação TDN na forma de hipertexto.

5.1.2. A Notação HTDN

Neste trabalho optou-se por utilizar uma variação da notação TDN na forma de hipertexto, que denominamos "HyperText Design Notation" (HTDN). Assim como a notação TDN, a HTDN representa a estrutura modular (relação *usa*) e a árvore conceitual (relação *é-composto-por*). Como é possível notar na figura 5.7, a estrutura modular é representada como na notação TDN, mas com a diferença de que cada módulo na lista da relação usa, é na verdade um botão que aciona um link para a posição no texto onde está definido aquele módulo. Além disso, é possível navegar nos dois sentidos da relação usa, buscando os módulos que usam determinado módulo. Ao lado da expressão "*usado por*" há um botão que gera, dinamicamente, uma excursão pelos pontos do texto em que estão as definições dos módulos que usam o módulo em questão.

Sistema: CONTROLE DE RADAR

Cluster: SENSORES
descrição: Módulos responsáveis pelo controle dos sensores.

Módulo: DETECTA_RADAR
descrição: recebe sinais detectados pelo radar
interface: var cod_radar: integer; type freq_comum: array [1..10] of real
usa: **MONITORA_FREQ, COMPUTA_DIST, CONTROLA_MSG**

Módulo: MONITORA_FREQ
descrição: monitora a frequencia recebida
usa: **MOSTRA_MSG**
usado por: *[excursão pelos módulos: Detecta_Radar]*

Módulo: COMPUTA_DIST
descrição: computa a distância do sinal

Módulo: CONTROLA_MSG
descrição: controla as mensagens a enviar
usa: **MOSTRA_MSG, SOA_ALARME**

Cluster: MENSAGENS
descrição: Módulos responsáveis pela apresentação de mensagens.

Módulo: MOSTRA_MSG
descrição: mostra mensagem na tela
usado por: *[excursão pelos módulos: Controla_Msg e Monitora_Freq]*

Módulo: SOA_ALARME
descrição: faz soar o som do alarme
usado por: *[excursão pelos módulos: Controla_Msg]*

Figura 5.7: Notação HTDN. Os nomes de módulo em negrito são links que transportam o leitor para a posição no texto onde estes módulos estão definidos.

Na notação HTDN, a árvore conceitual (relação *é-composto-por*) pode ser visualizada através da endentação dos níveis de *clusterização*. No exemplo da figura 5.1 é possível perceber três níveis: sistema, cluster e módulo, onde sistema e módulo são, respectivamente, os níveis máximo e mínimo de abstração. No entanto, nada impede que se criem níveis de abstração intermediários.

Uma vantagem interessante da HTDN é que com os recursos de abstração oferecidos pelos sistemas de hipertexto, é possível ajustar a visualização da árvore conceitual, conforme a necessidade do leitor. Caso este deseje visualizar apenas os níveis mais altos, basta que ajuste a sua visão, escondendo os níveis que não lhe interessam, como mostra a figura 5.8.

Sistema: CONTROLE DE RADAR

Cluster: SENSORES
descrição: Módulos responsáveis pelo controle dos sensores.

Cluster: MENSAGENS
descrição: Módulos responsáveis pela apresentação de mensagens.

Figura 5.8: Visualização com abstração.

5.2. O Processo de Recuperação do Projeto Arquitetônico

A partir das definições de Recuperação de Projeto apresentadas anteriormente pode-se distinguir nesta atividade duas tarefas. A primeira tarefa é a análise do sistema propriamente dito e identificação de seus módulos e inter-relacionamentos. A segunda, onde a participação humana é imprescindível, corresponde à aquisição do conhecimento dos especialistas no sistema em questão.

No ambiente proposto, estas duas tarefas são executadas em duas etapas. A primeira é a etapa automática, responsável pela análise do sistema e obtenção de sua estrutura modular e árvore conceitual. A etapa cooperativa é a segunda, e é responsável pela obtenção do conhecimento dos especialistas. As sessões seguintes explicam cada etapa do processo de Recuperação de Projeto Arquitetônico, e comentam quais as características necessárias às ferramentas de apoio a este processo.

5.2.1. Etapa Automática

Muitas vezes, a única informação que se tem de um sistema é o seu código fonte. Algumas organizações possuem documentações associadas ao código, mas dificilmente

garantem a sua atualização em relação ao mesmo. Tomando a situação mais comumente encontrada, consideramos o código fonte como a única fonte de informação confiável e disponível sobre o sistema, para proceder a etapa automática.

A etapa automática envolve quatro tarefas: a identificação dos módulos do sistema, o estabelecimento das relações existentes entre estes módulos, a extração de sua interface e descrição, e a geração destas informações em HTDN. O ambiente de programação, a linguagem usada e os hábitos dos programadores do sistema, são fatores que influenciam fortemente as três primeiras tarefas. Já a quarta tarefa depende apenas do resultado obtido com a execução das outras três.

✂ *Identificando os módulos*

Uma mesma organização pode apresentar diversos ambientes de programação, cada qual com suas próprias diretrizes e características. O ambiente e a linguagem de programação são os responsáveis pela organização dos fontes e pela identificação do que se pode considerar como módulo. Em geral, os módulos de um sistema correspondem aos arquivos de programas fonte que o compõem. Por exemplo, nos sistemas escritos em COBOL cada arquivo é um programa que corresponde a um módulo. Por outro lado, os sistemas escritos em Turbo Pascal utilizam o conceito de bibliotecas, onde em um único arquivo residem várias rotinas que podem ser identificadas como módulos. Neste caso, os arquivos fonte devem ser analisados internamente. Outro aspecto a considerar, é a utilização de "includes" na programação em C e em Pascal, ou de "copy's" em COBOL. Este tipo de prática leva à identificação de outros tipos de módulo, que também comporão a estrutura modular do sistema.

❄ *Identificando os relacionamentos*

As relações *usa* e *é-composto-por* devem ser identificadas. De maneira geral, a relação *usa* entre os módulos é determinada pela "chamada" que um programa faz do outro, isto é, quando um programa Detecta_Radar "chama" o programa Computa_Distância, isso significa que Detecta_Radar *usa* Computa_Distância, e que portanto, existe um relacionamento entre estes dois módulos. Da mesma forma, quando um programa "menciona" um arquivo, isto significa que o tal programa *usa* o tal arquivo.

A identificação da relação *é-composto-por* depende da forma de *clusterização* que se quer adotar. Como já foi citado anteriormente, os sistemas costumam estar *clusterizados* segundo algum critério, seja este ideal ou não. Existem algorítmos de *clusterização*, como o proposto por Schwanke e Platoff [ScPl89], que poderiam ser utilizados na geração de uma *clusterização* melhor. No entanto, não convém impor uma nova clusterização, mas sim, sugerir mudanças na existente. A identificação da clusterização existente varia de acordo com o ambiente e linguagem de programação usados. No ambiente COBOL encontra-se a clusterização embutida no nome do fonte ou do diretório em que este se encontra. Já no ambiente Turbo Pascal podemos identificar *clusters* na forma de "units".

Extraindo a Interface e a Descrição

A interface entre módulos é aquilo que é preciso saber para *usar* um módulo. Normalmente, os parâmetros esperados e retornados de e por um módulo são identificados como sua interface. Em relação à descrição dos módulos, se o objetivo é obter algum texto em linguagem natural, vamos depender da qualidade e utilização de comentários nos fontes. Quanto mais descritivos forem os nomes de variáveis e módulos, e quanto mais comentários elucidativos acompanharem a descrição do programa, melhor será o resultado do trabalho de Recuperação de Projeto.

Gerando a HTDN

Uma vez terminadas as tarefas anteriores, o que nos resta é organizar os dados obtidos no formato da HTDN. Para tanto, é necessário contar com uma ferramenta de hipertexto que ofereça uma interface para importação de textos comuns. Algumas ferramentas de hipertexto fornecem uma linguagem de marcação de texto ("mark-up language"), com a qual é possível adaptar um texto comum e depois, transportá-lo automaticamente para dentro do ambiente da ferramenta. Desta forma, os links necessários à representação da estrutura modular, e os níveis necessários à representação da árvore conceitual podem ser criados ainda nesta etapa.

Uma ferramenta de apoio a esta etapa deve cobrir as quatro tarefas descritas acima, levando em conta, não só o ambiente de programação em que se encontra o sistema,

como também, a ferramenta de hipertexto para a qual se quer transportar a informação sobre o sistema.

5.2.2. Etapa Cooperativa

A etapa automática dificilmente consegue extrair dos códigos fonte uma descrição completa de cada módulo, baseando-se somente nos comentários encontrados. O enriquecimento da descrição de cada módulo depende do apoio dos especialistas no sistema, que em geral são os membros da equipe responsável por sua manutenção. O envolvimento deste grupo de pessoas caracteriza a segunda etapa como cooperativa.

Sabe-se que a etapa automática gera uma documentação na forma hipertextual, contando com a interface de uma ferramenta de hipertexto, que permite a vizualização da informação recuperada. Mas além desta característica os sistemas de hipertexto oferecem outras características que favorecem a segunda etapa da Recuperação do Projeto Arquitetônico. Fletton e Munro [FlMu88] identificam estas características como ideais para uma ferramenta de apoio à Redocumentação de sistemas. Acreditamos que estas mesmas características aplicam-se às ferramentas de apoio à Recuperação de Projeto.

→ Incremental: Como o conhecimento dos especialistas encontra-se disperso e não formatado, leva-se tempo para extraí-lo. Além disso, a equipe de manutenção tem como atividade principal efetuar as mudanças necessárias ao sistema, sendo a tarefa de Recuperação quase sempre colocada em segundo plano. Portanto, a ferramenta deve oferecer meios de aquisição e organização gradual da informação.

→ Atualização Casual: Os especialistas devem ter um fácil acesso à ferramenta, de forma que a extração de seu conhecimento seja o mais casual possível. Portanto, é importante que a ferramenta esteja próxima ao ambiente de trabalho dos especialistas.

→ Garantia de Qualidade: A ferramenta deve prover mecanismos de controle da qualidade da informação adquirida, verificando e confirmando as fontes da mesma.

→ Trabalho Cooperativo: Uma vez que existe um grupo de pessoas interessadas e detentoras da informação sobre o sistema, a ferramenta deve permitir que todos possam cooperar, seja para oferecer como para receber informação. Aspectos como o comportamento do grupo, característica dos membros, e a atividade que se quer apoiar, devem ser cuidadosamente estudados e analisados, com o objetivo de construir uma ferramenta adequada.

→ Gerência de Configuração: A ferramenta deve acompanhar e manter versões da informação de acordo com a versão do sistema a que se associa.

→ Integração ao Código Fonte: O código fonte deve ser facilmente alcançável a partir da informação apresentada pela ferramenta.

→ Integração com ferramentas automáticas: Informações geradas por outras ferramentas automáticas devem ser facilmente acessíveis e importáveis pela ferramenta.

→ Ocultamento de Informação: A ferramenta deve prover níveis de abstração opcionais, de modo que facilite ao usuário o entendimento da documentação por ela apresentada.

Assim como ocorre na fase de Projeto durante a engenharia progressiva, o processo reverso também exige que os membros da equipe discutam entre si. Esta interação ajuda os projetistas, ou recuperadores de Projeto, a gerar um documento de melhor qualidade. E mais, o registro desta discussão também é importante, pois um documento contendo toda a discussão ocorrida na etapa cooperativa, mais tarde poderá facilitar o entendimento de como se originou o documento de Projeto. Assim sendo, além das características identificadas acima, a ferramenta de apoio a esta etapa deve incluir mecanismos de captura da discussão.

No capítulo 4 comentamos sobre alguns dos modelos usados para capturar a argumentação. Acreditamos que por sua forma natural e intuitiva de expressão [ReEl91], o modelo IBIS pode ser usado para apoiar a equipe de manutenção durante as discussões de recuperação de Projeto. No entanto, durante a interação assíncrona dos membros da equipe, não é conveniente que a argumentação seja anônima. Como o modelo IBIS não se preocupa com a identificação dos argumentadores, para supri-la propomos que toda a ocorrência de uma questão, proposta ou argumento, inclua o nome daquele que a originou.

Os documentos, Projeto Arquitetônico e Discussão de Projeto, estão fortemente relacionados, portanto, ambos precisam estar o mais próximo possível. Para manter esta proximidade, a mesma ferramenta de hipertexto que armazena o documento de Projeto, deve ser capaz de armazenar o documento da Discussão. Os sistemas de hipertexto são conhecidos por seu grande potencial de conecção entre documentos, fazendo crer que é possível prover esta proximidade. Mas além disso, o sistema de hipertexto escolhido, deve permitir a implementação do modelo IBIS, apoiando não só

a co-edição do documento de Projeto, como também o processo de discussão sobre o mesmo. O modelo IBIS já foi implementado graficamente em um sistema de hipertexto conhecido como gIBIS [CoBe87], entretanto, as facilidades oferecidas pelo sistema de hipertexto escolhido para apoiar a etapa cooperativa, é que determinarão a forma de implementação deste modelo.

A escolha ou construção de ferramentas de apoio às duas etapas que formam um Ambiente de Recuperação de Projeto, deve ser cuidadosa, buscando sempre o preenchimento da maior parte dos requisitos necessários, aqui comentados.

Capítulo 6
O Protótipo ARCoPAS

O ARCoPAS é a implementação de um protótipo de Ambiente para a Recuperação Cooperativa do Projeto Arquitetônico de Sistemas. Como foi comentado anteriormente, o ambiente proposto neste trabalho, divide a tarefa de recuperação em duas etapas: automática e cooperativa. Para apoiar a primeira etapa foi desenvolvida uma ferramenta chamada ExEM. Ela é a responsável pela análise dos fontes: identificando os módulos e as suas interrelações, e gerando um texto com a estrutura modular e a árvore conceitual no formato adequado para etapa seguinte.

Para apoiar a etapa cooperativa, utiliza-se a ferramenta FolioViews[1] em conjunto com a notação itIBIS. O FolioViews é um sistema de hipertexto, através do qual é possível visualizar e complementar o documento de Projeto, assim como "discutir" sobre a sua complementação. Para capturar esta discussão segundo o modelo IBIS, utiliza-se a notação itIBIS, implementada sobre o FolioViews. A figura 6.1 apresenta uma esquema global do ARCoPAS, identificando suas etapas e ferramentas de apoio.

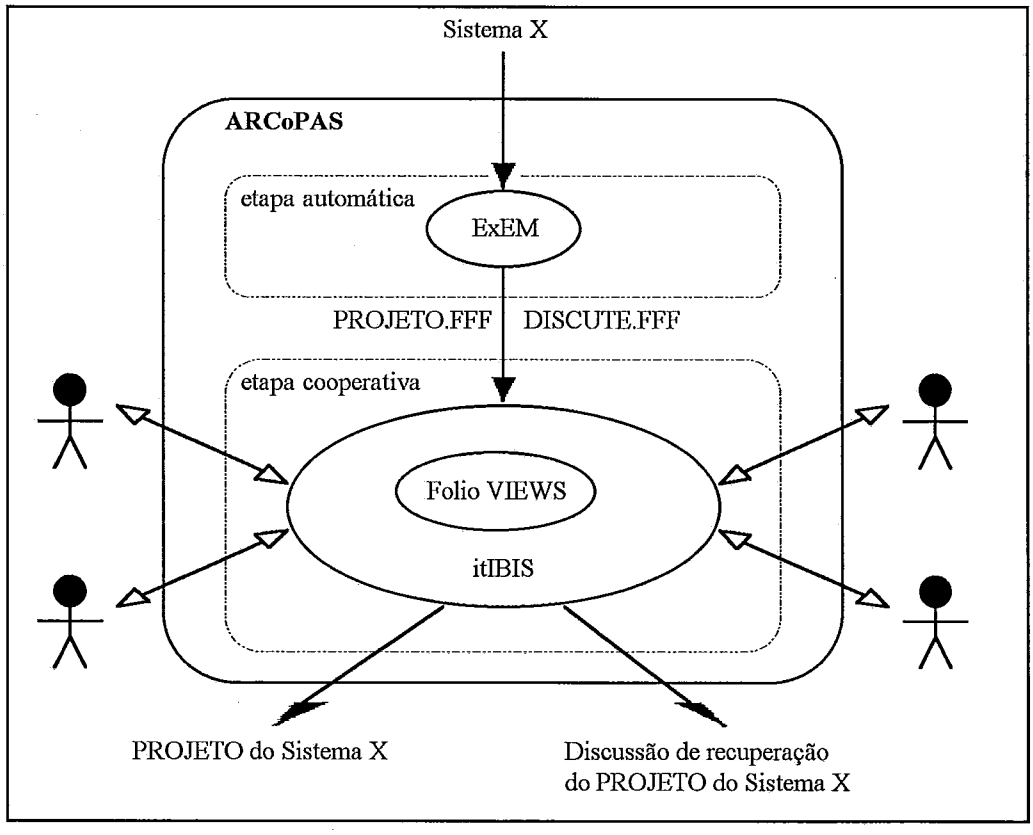


Figura 6.1: Esquema do ambiente ARCoPAS.

[1] FolioViews é marca registrada da Folio Corporation

6.1. A ferramenta ExEM

A ferramenta ExEM **ex**trai a **e**strutura **m**odular de um sistema, sendo a ponte entre os fontes do mesmo e o hiperdocumento. É ela a responsável pela identificação dos módulos e relacionamentos do sistema, assim como pela geração de um hipertexto com a estrutura modular e a árvore conceitual do mesmo utilizando a notação HTDN.

Em uma primeira versão, a ExEM foi desenvolvida para analisar fontes escritos na linguagem de programação Turbo Pascal. Os sistemas escritos nesta linguagem costumam ter dois tipos de arquivo fonte: *programs* e *units*. O tipo *program* dispensa explicação, entretanto é preciso esclarecer o que vem a ser uma *unit*. Segundo o manual do usuário do Turbo Pascal 7.0, uma *unit* é uma coleção de constantes, tipos de dados, variáveis, procedimentos e funções [Borl93].

Um sistema típico em Turbo Pascal é composto por um fonte do tipo programa e diversos fontes do tipo *unit*. Para extrair a estrutura modular de um sistema deste tipo é preciso estabelecer qual a granularidade desejada. De acordo com o granularidade escolhida, um módulo pode ser desde um arquivo fonte até uma variável. No entanto, como o objetivo deste trabalho é prover a recuperação do Projeto Arquitetônico, buscamos um nível alto de abstração. A ExEM identifica como módulos, os procedimentos e funções (públicos e privados) dos fontes do tipo *unit*, e os fontes do tipo programa. Para obter o nível de abstração ideal, a ExEM desconsidera variáveis e procedimentos internos a outros procedimentos.

Os procedimentos e funções que compõem uma *unit*, podem *usar* uns aos outros. Para descobrir a relação *usa* entre eles, a ExEM varre cada um, procurando por referências aos demais. Para determinar a relação *usa* entre módulos de diferentes *units* e, entre estes e o *program*, a ExEM baseia-se na cláusula "uses". Nesta cláusula há uma lista das *units* que podem ser *usadas* pela *unit* ou pelo *program* que as declarou. O fato de um *unit* A poder usar a *unit* B, significa que os procedimentos e funções que compõem a *unit* A podem usar os procedimentos e funções públicos que compõem a *unit* B. Entretanto, nem todos os procedimentos e funções são realmente usados. A ExEM desdobra as listas declaradas na cláusula "uses", gerando a real relação *usa* entre os procedimentos e funções.

A relação *é-composto-por* (árvore conceitual) também é identificada pela ExEM. Considera-se cada *unit* como um *cluster*, que representa todos os módulos públicos e privados da *unit*. A ExEM assume a *clusterização* existente, incluindo na identificação de cada módulo, a *unit* a que pertence.

Quanto à descrição de um módulo, para que a ExEM pudesse extraí-la automaticamente, seria preciso reconhecer o formato padrão utilizado pelos programadores. Em geral, cada ambiente de programação tem suas próprias regras, tornando a tarefa de padronização muito difícil. Desta forma, fomos obrigados a estabelecer um formato que a ExEM reconhecesse. Caso o módulo possua alguma descrição, esta deve ser encontrada na linha imediatamente após a declaração do mesmo, iniciando-se com a seguinte sequência de caracteres: '*{descrição:* '. Não encontrando a descrição neste formato, a ExEM extrai os comentários encontrados no trecho correspondente ao módulo para facilitar o trabalho na etapa cooperativa.

```
unit A;
interface
   uses B;

   procedure A1;
   procedure A2;

implementation
procedure A1;
        ...
        B1;
        B2;
        ...
procedure A2;
      ..
      A3;.
      B1;
      ...
procedure A3;
      ...
      B2;
      ...
```

Figura 6.2a: Fonte A.PAS

```
unit B;
interface
    procedure B1
        (var S: string);
    procedure B2;

implementation
procedure B1;
{descrição: reverte o
conteúdo de S }
     ...
procedure B2;
     ...
procedure B3;
                • • •
```

Figura 6.2b: Fonte B.PAS

Como já foi dito anteriormente a interface de um módulo são os parâmetros necessários para a permitir a sua utilização por outro módulo. A ExEM encontra a interface de cada módulo logo após a declaração dos mesmos, não sendo necessário fazer qualquer transformação.

Para ilustrar o funcionamento da ExEM, tomemos como exemplo um sistema S. Sejam A e B duas *units* pertencentes ao sistema S. Sejam A1 e A2 dois procedimentos públicos de A, A3 um procedimento privado de A, B1 e B2 dois procedimentos públicos de B, e B3 um procedimento privado de B. As figuras 6.2a e 6.2b apresentam um esboço do conteúdo dos arquivos fonte que contém as *units* A e B. A figura 6.3 apresenta o arquivo gerado pela ExEM com a descrição dos módulos identificados a partir dos fontes analisados, usando a notação HTDN.

No exemplo apresentado no figura 6.3, os botões estão entre colchetes. Como já comentamos anteriormente, em uma ferramenta de hipertexto, os botões transportam o leitor para um outro ponto do documento que se está navegando. Na definição de cada módulo podem haver quatro tipos de botão: programa, usa, excursão e discussão. O botão *programa* transporta o leitor para o ambiente de programação Turbo Pascal, trazendo para edição o programa fonte correspondente àquele módulo. Os botões do tipo *usa* correspondem aos nomes de

módulos que estão em negrito. Ao acionar o botão B.B2 que consta na lista de módulos usados pelo módulo A.A1, o leitor é levado à descrição do módulo B.B2. O botão *excursão* cria dinamicamente uma excursão pelos módulos que são usados por aquele módulo. E finalmente, o botão *discussão* transporta o leitor para a área em que lhe é permitido discutir sobre a descrição daquele módulo.

A ferramenta de hipertexto utilizada pelo ARCoPAS, o FolioViews, provê uma linguagem de marcação de texto que permite que um texto em ASCII possa ser importado para seu ambiente. Utilizando estas marcações, é possível informar ao FolioViews onde se quer criar os botões e para onde estes devem transportar o leitor, possibilitando desta forma, a geração automática dos links. A seção seguinte descreve o FolioViews e explica como é gerado o hiperdocumento no formato adequado a esta ferramenta.

Sistema S

Cluster A [programa] [discussão]
descrição:
Módulo A.A1 [programa] [discussão]
descrição:
usa: ***[B.B1], [B.B2]***

Módulo A.A2 [programa] [discussão]
descrição:
usa: ***[B.B1], [A.A3]***

Módulo A.A3 [programa] [discussão]
descrição:
usa: ***[B.B2]***
é usado por: [excursão para A.A2]

Cluster B [programa] [discussão]
descrição:
Módulo B.B1 [programa] [discussão]
descrição: reverte o conteúdo de S
interface: var S: string
é usado por: [excursão para A.A1, A.A2]

Módulo B.B2 [programa] [discussão]
descrição:
usa: ***[B.B3]***
é usado por: [excursão para A.A1, A.A3]

Módulo B.B3 [programa] [discussão]
descrição:
é usado por: [excursão para B.B2]

Figura 6.3: Descrição do sistema S na notação HTDN

Uma descrição ainda mais detalhada e completa da ferramenta ExEM é dada por seu Projeto Arquitetônico, que foi extraído utilizando-se o próprio ARCoPAS. Uma versão textual deste documento pode ser encontrada no Apêndice.

6.2. A ferramenta FolioVIEWS

Uma ferramenta de hipertexto adequada à segunda etapa da Recuperação de Projeto não seria fácil desenvolver a partir do zero. Optou-se então por utilizar um software que melhor se adaptasse às características necessárias ao ambiente proposto. A

ferramenta escolhida é um co-editor de hipertextos [Foli93] chamado FolioVIEWS[2] versão 3.0, que pode ser classificado, dentro da taxonomia proposta por Olson et al. [Olso93], como um sistema de co-autoria.

O FolioViews armazena e gerencia seus hipertextos em arquivos denominados Infobases. Uma Infobase é composta por registros, que constituem a menor unidade de informação manipulável. Os registros podem ser agrupados e/ou associados a níveis hierárquicos. Os grupos e níveis gerados por esta associação, permitem a organização e estruturação da informação na Infobase.

Uma grande facilidade do FolioViews é a sua capacidade de efetuar consultas na Infobase. Cada palavra da Infobase é indexada, permitindo maior eficiência e rapidez nas consultas. Uma consulta ou "querie" gera uma excursão pela infobase, com passagem por todas as ocorrências encontradas. A organização do conteúdo de uma Infobase em grupos e níveis acrescenta à facilidade de consulta, um poder ainda maior, permitindo que as consultas possam se restringir a determinado grupo ou nível.

O FolioViews provê cinco tipos de link, que permitem ao leitor navegar dentro de uma ou mais Infobases. Os *jump links* transportam o leitor de um ponto a outro da Infobase. Os *pop-up links* mostram uma pequena janela onde pode haver textos ou gráficos. Os *object links* abrem uma janela onde haverá um gráfico do tipo Bitmap ou Windows Metafile. *Program links* iniciam a execução de uma aplicação externa. E finalmente, há os *Query links,* que provocam a execução de uma consulta.

A característica de co-edição do FolioViews é reforçada pelas facilidades de anotação e marcação de texto oferecidas por ele. O usuário pode anotar observações em uma área separada e vinculá-la a um parágrafo do texto ("notes"). Outro recurso é a possibilidade de utilizar marcadores de texto ("highlighters") com diferentes formatos, que podem ser definidos pelo próprio usuário.

O FolioViews é uma ferramenta multi-usuário. Ele suporta até 125 usuários autores acessando uma única Infobase ou cópias individuais da mesma, simultaneamente, para editar, anotar ou marcar seu conteúdo. Por outro lado, um número ilimitado de usuários pode usar simultaneamente uma mesma Infobase, no modo de leitura. Uma Infobase pode estar ou não disponível para múltiplos usuários. Esta condição é definida no momento da abertura de uma Infobase, quando o usuário pode torná-la privada ao desligar a opção multi-usuário ("multi-user").

Quando múltiplos usuários estão acessando e modificando uma Infobase

[2]Folio Views é marca registrada da Folio Corporation.

simultaneamente, o FolioViews "prende" o registro em alteração, ao usuário que o está alterando. Durante a atualização deste registro, todos os demais usuários poderão apenas visualizá-lo, recebendo um aviso quando tentam modificá-lo. Após salvar as mudanças feitas, o registro modificado é liberado para outras alterações.

A cópia individual de uma Infobase ("shadow file") permite ao usuário ter um acesso particular à mesma. Nesta cópia o usuário pode anotar, marcar, mudar estilos e editar o conteúdo da Infobase sem precisar compartilhar estas alterações com os demais usuários da mesma. Cada Infobase pode ter um número ilimitado de cópias individuais, permitindo que um grupo de pessoas trabalhe sobre uma mesma Infobase simultaneamente, e mantenha em separado suas anotações pessoais.

Para fins de segurança, o FolioViews provê um controle de acesso às suas Infobases. Além de permitir a associação de senhas a usuários individuais ou a grupos de usuário com relação a uma Infobase, é possível ainda, definir que direitos terão os usuários de cada senha. Por exemplo, é possível negar o direito de edição e permitir o direito de anotação e marcação do texto, a determinado grupo de usuários de uma Infobase.

Como foi dito anteriormente o FolioViews oferece uma linguagem de marcação de textos que permite a importação de textos comuns para o formato de suas Infobases. A ferramenta ExEM gera arquivos para a criação de duas Infobases no ambiente do FolioViews: Projeto e Discute. Para cada Infobase a ser criada, a ExEM gera dois arquivos, um com a informação que constituirá a Infobase (.FFF) e outro com a definição dos estilos (formatos), níveis e objetos utilizados na estruturação desta informação (.DEF).

As figuras 6.4 e 6.5 mostram como ficam os arquivos PROJETO.DEF e PROJETO.FFF gerados pela ExEM a partir da análise de arquivos fonte escritos em Turbo Pascal, de um sistema chamado CIRCO.

```
<UG:analista,Owner>
<UG:programador,Print,Bookmark,Note,M_Pen,M_Group,Save_As>
<MS:analista,carlo,manoel>
<MS:programador,giga>
<UR:carlo,carlo>
<UR:manoel,manoel>
<UR:giga,giga>
<LN:sistema,cluster,modulo,interface,usa>
<LE:"Normal Level","Normal Level",IN:0.5,0,0>
<LE:sistema,sistema,JU:CN,BP:0.5,BD+,FT:Helv,PT:18>
<LE:cluster,cluster,BP:0.25,IN:0.25,0,-0.249306,BD+,PT:14>
<LE:modulo,modulo,BP:0.125,IN:0.5,0,-0.248611,BD+>
<LE:interface,interface,IN:1.15,0,-0.65>
<LE:usa,usa,IN:0.80,0,-0.30>
<CL:Bitmap>
<OD:editor,Bitmap,"C:\CIRCO\editor.BMP">
<OD:usado,Bitmap,"C:\CIRCO\usado.BMP">
<OD:discute,Bitmap,"C:\CIRCO\discute.BMP">
<ST:Program,LK,FC:128,0,0>
<ST:Query,LK,FC:128,0,128>
```

Figura 6.4: Arquivo PROJETO.DEF gerado para o sistema CIRCO.

O arquivo PROJETO.DEF mostrado na figura 6.4, contém diversas definições. Cada linha do arquivo encontra-se delimitada pelos caracteres "<" e ">" e inicia-se por um código com duas letras maiúsculas, significando o que será definido a seguir. O código UG define os grupos de usuários da Infobase e quais os direitos ou ações permitidas a estes grupos. Os códigos UR e MS definem, respectivamente, os usuários da Infobase e a qual grupo pertencem ("MemberShip"). Os níveis que estruturarão o conteúdo da Infobase e seus respectivos estilos (formatos), são definidos pelos códigos LN e LE. O código OD define os objetos do tipo bitmap utilizados pela Infobase. E finalmente, o código ST define o estilo de dois tipos de link oferecidos pelo FolioViews: *program link* e *query link*.

Um trecho do arquivo PROJETO.FFF é ilustrado pela figura 6.5. Também neste arquivo, as marcações aparecem precedidas por códigos com duas letras maiúsculas e delimitadas pelos caracteres "<" e ">". A primeira linha contém o código DI, que indica que o nome a seguir, corresponde ao arquivo de definições (".DEF"). Estas definições estabelecem estilos e características, relativas às marcações existentes no arquivo PROJETO.FFF. Cada registro da InfoBase é definido depois de um código RD, seguido do nome do nível, definido no arquivo ".DEF", que está associado àquele registro. Os códigos QL e PL indicam o início da área onde existe um botão ativador de um link, sendo o código EL usado para indicar o término desta área. E por último, o código OB indica a existência de um objeto.

```
<DI:"projeto.DEF">
<RD:sistema>Projeto Arquitetonico do Sistema CIRCO
<RD:Cluster>Cluster: AREAS
  <PL:program,"d:\bp\bin\bp C:\CIRCO\AREAS.pas">
    <OB:editor,Bitmap><EL>
  <QL:Query,"Cluster AREAS","C:\CIRCO\discute.NFO">
    <OB:discute,Bitmap><EL>
<RD>descricao:

<RD:Modulo>Modulo: AREAS.MUDA_VIS
  <PL:program,"d:\bp\bin\bpC:\CIRCO\AREAS.pas">
    <OB:editor,Bitmap><EL>
  <QL:Query,"ModuloAREAS.MUDA VIS","C:\CIRCO\discute.NFO">
    <OB:discute,Bitmap><EL>
<RD>descricao:
<RD:interface>interface: (oldvis: tipomasc; vem_do_mouse:boolean)
<RD:usa>usa:
  <QL:Query,"Modulo GRAF.APAREDESENHA">
     GRAF.APA_REDESENHA<EL>,
  <QL:Query,"Modulo GRAF.POE CUR">GRAF.POE_CUR<EL>,
  <QL:Query,"Modulo GRAF.TIRACUR">GRAF.TIRA_CUR<EL>
<RD>usado por: <QL:Query,"[level usa:AREAS.MUDA VIS]",RH>
  <OB:usado,Bitmap><EL>

<RD:Modulo>Modulo: AREAS.MOVE_JAN
  <PL:program,"d:\bp\bin\bp C:\CIRCO\AREAS.pas">
    <OB:editor,Bitmap><EL>
  <QL:Query,"Modulo AREAS.MOVE JAN","C:\CIRCO\discute.NFO">
    <OB:discute,Bitmap><EL>
<RD>descricao:
<RD:usa>usa: <QL:Query,"Modulo GRAF.APA REDESENHA">
     GRAF.APA_REDESENHA<EL>,
  <QL:Query,"Modulo GRAF.DESENHA NOS">GRAF.DESENHA_NOS<EL>,
  <QL:Query,"Modulo GRAF.MOSTRA CONEXOES">
     GRAF.MOSTRA_CONEXOES<EL>,
  <QL:Query,"Modulo GRAF.POE CUR">GRAF.POE_CUR<EL>,
  <QL:Query,"Modulo GRAF.SCROLL TELA">GRAF.SCROLL_TELA<EL>,
  <QL:Query,"Modulo GRAF.TIRA CUR">GRAF.TIRA_CUR<EL>,
  <QL:Query,"Modulo GRAF.VIEW PORT">GRAF.VIEW_PORT<EL>
<RD>usado por: <QL:Query,"[level usa:AREAS.MOVE JAN]",RH>
  <OB:usado,Bitmap><EL>
```

Figura 6.5: Trecho do arquivo PROJETO.FFF gerado para o sistema CIRCO.

Uma vez dentro do ambiente do FolioViews, no momento em que o usuário solicitar a "abertura" do arquivo PROJETO.FFF, utilizando o tipo "Folio Flat File", a conversão é disparada automaticamente. Ao final da conversão, o usuário passa a manipular um arquivo chamado PROJETO.NFO, que corresponde à Infobase com o documento de Projeto na notação HTDN. A figura 6.6 mostra, na forma textual, como fica um trecho da Infobase gerada a partir dos arquivos PROJETO.DEF e PROJETO.FFF.

Projeto Arquitetônico do Sistema CIRCO

Cluster: AREAS 
descrição:

Módulo: AREAS.MUDA_VIS 
descrição:
interface: (oldvis: tipomasc; vem_do_mouse:boolean)
usa: **GRAF.APA_REDESENHA, GRAF.POE_CUR, GRAF.TIRA_CUR**
usado por: 

Módulo: AREAS.MOVE_JAN 
descrição:
usa: **GRAF.APA_REDESENHA, GRAF.DESENHA_NOS,**
GRAF.MOSTRA_CONEXOES, GRAF.POE_CUR,
GRAF.SCROLL_TELA, GRAF.TIRA_CUR, GRAF.VIEW_PORT
usado por: 

Figura 6.6: Trecho extraído da Infobase PROJETO.NFO gerada para o sistema CIRCO.

Os ícones que aparecem na figura 6.6 são botões que acionam os seguintes links previamente preparados pela ExEM:

 - *program link* que dispara o editor de Turbo Pascal da Borland com o fonte correspondente ao módulo ou cluster em que o usuário está posicionado;

 - *query link* com uma consulta restrita a Infobase DISCUTE, para encontrar a área de discussão sobre o cluster ou módulo em que o usuário está posicionado;

 - *query link* com uma consulta restrita ao nível **usa**, para obter os registros com os módulos que usam o módulo em que o usuário está posicionado.

Ainda na figura 6.6, nota-se que a lista de módulos usados está em negrito. Esta evidenciação se deve ao fato de que o nome de cada módulo na lista, é na verdade um *query link* com uma consulta, que leva à definição correspondente àquele módulo na mesma Infobase.

O FolioViews é muito parecido com os editores de texto conhecidos, o que torna a sua utilização bastante confortável para os integrantes do grupo de Recuperação. Mas além da edição, o FolioViews oferece diversas outras facilidades relativas à sua característica de hipertexto. O aproveitamento destas facilidades exige que a equipe que irá utilizá-lo receba um treinamento adequado. Ilustraremos a seguir algumas das

facilidades mais interessantes, através de exemplos na Infobase Projeto.

As figuras 6.7, 6.8 e 6.9 mostram a formulação de uma consulta ("query") e o resultado fornecido pela mesma. A consulta apresentada por estas figuras corresponde a uma verificação da descrição de todos os módulos de um dado cluster. Note que ao formulá-la, o usuário deve restringir o escopo da busca ao nível de "cluster", e especificar qual o cluster desejado (veja a figura 6.7). Para visualizar somente os registros da Infobase com as descrições, basta mudar a Visão da mesma, escolhendo a opção "Records with Hits", como mostra a figura 6.8.

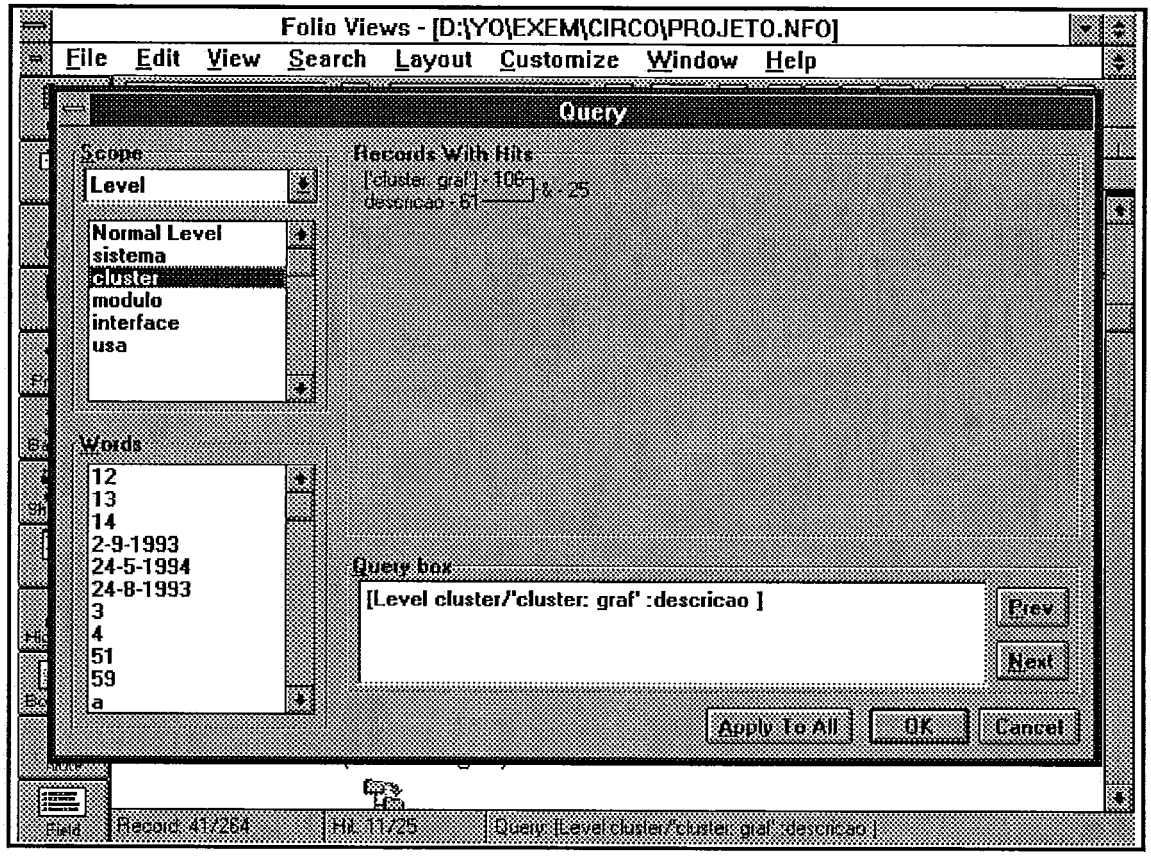


Figura 6.7: Formulação de uma consulta.

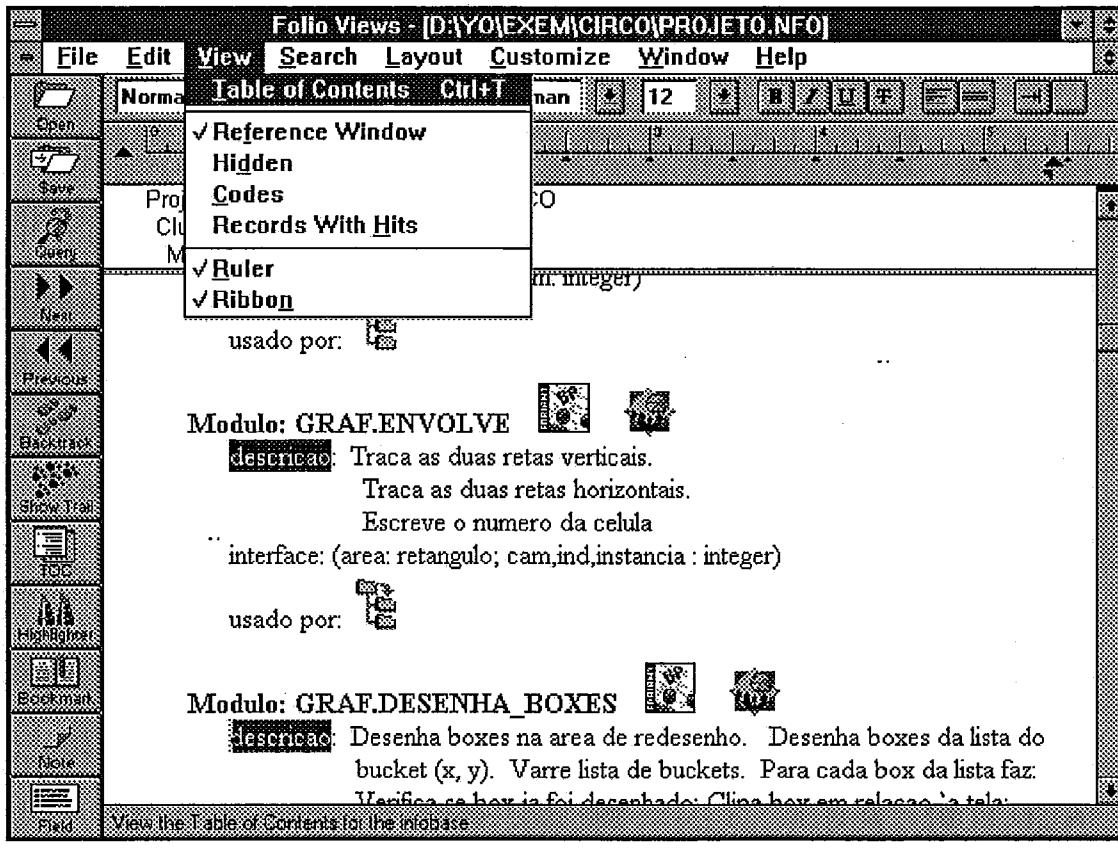


Figura 6.8: Mudando a visão da Infobase.

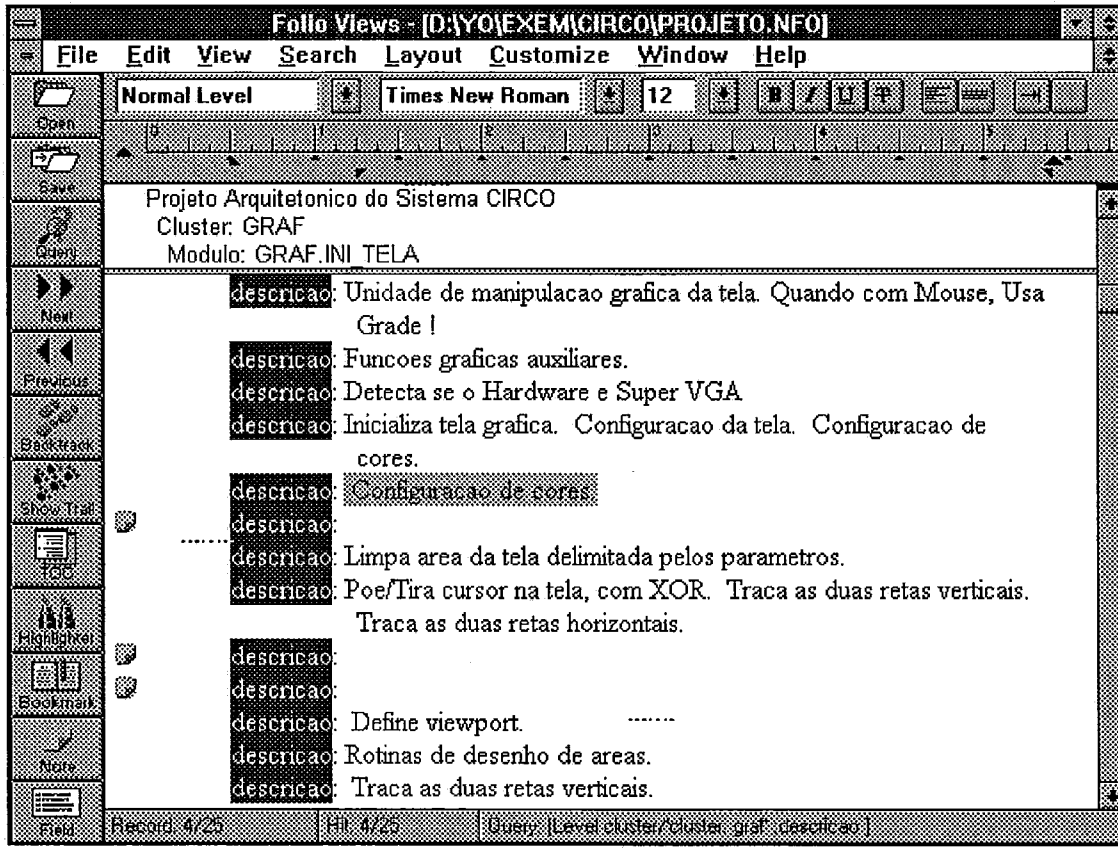


Figura 6.9: Resultado da consulta.

Uma consulta no FolioViews gera uma excursão pelos registros da Infobase onde foram encontradas ocorrências do que se consultou. A palavra "descrição" foi o objeto consultado no exemplo da figura 6.7, e cada ocorrência desta palavra está evidenciada como mostra a figura 6.9. Os botões **Next** e **Previous** situados na barra de botões no lado esquerdo da tela, ao serem acionados percorrerão os registros pertencentes à excursão gerada.

Ainda na figura 6.9, podemos observar a existência de anotações e marcações no texto. Para criar uma anotação relativa a um registro, basta posicionar-se no registro desejado e acionar o botão **Note** situado na barra de botões. Uma vez criada a anotação, um ícone característico aparece ao lado do registro em questão. Ao acionar este ícone, uma pequena janela se abre, mostrando o que foi anotado (veja a figura 6.10).

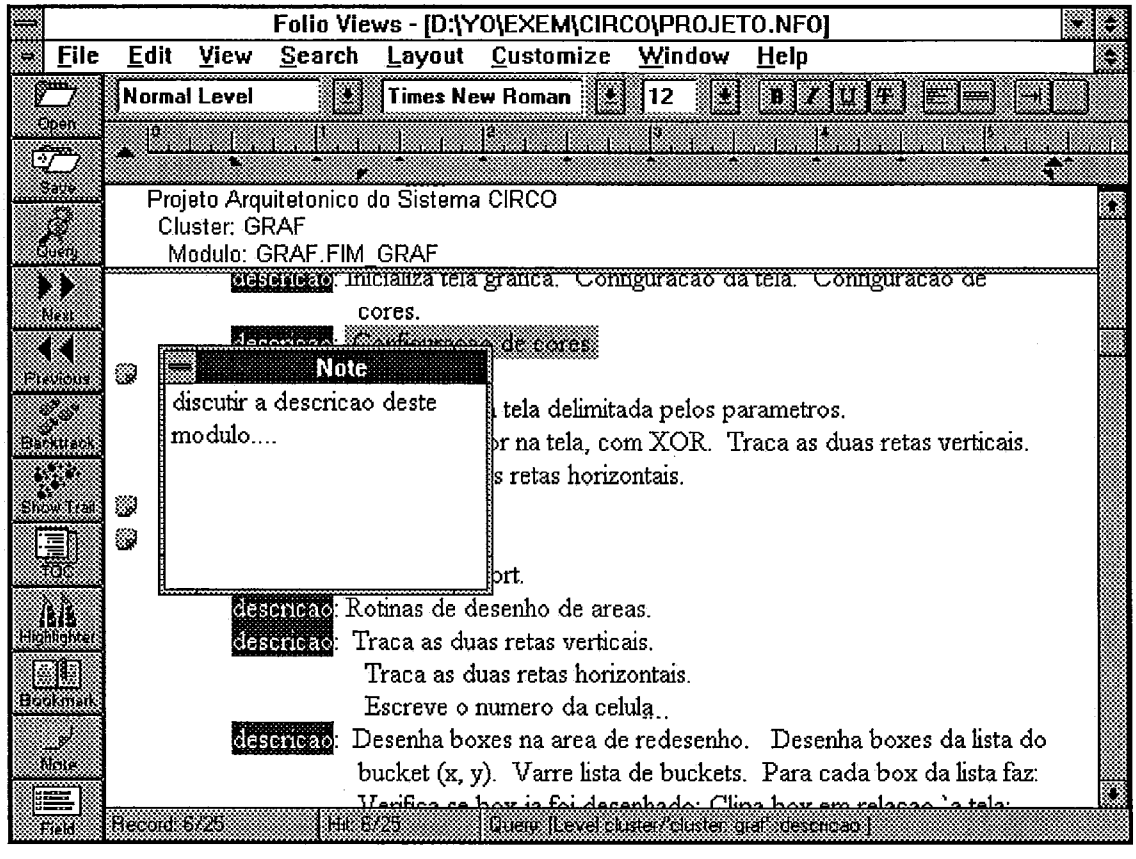


Figura 6.10: Acionando uma anotação.

Para marcar um trecho do texto da Infobase, basta selecionar o trecho desejado e acionar o botão **Highlighter** situado na barra de botões. O FolioViews permite que se definam vários tipos de marcações, assim o usuário deve escolher qual delas utilizará para marcar o texto selecionado (veja a figura 6.11).

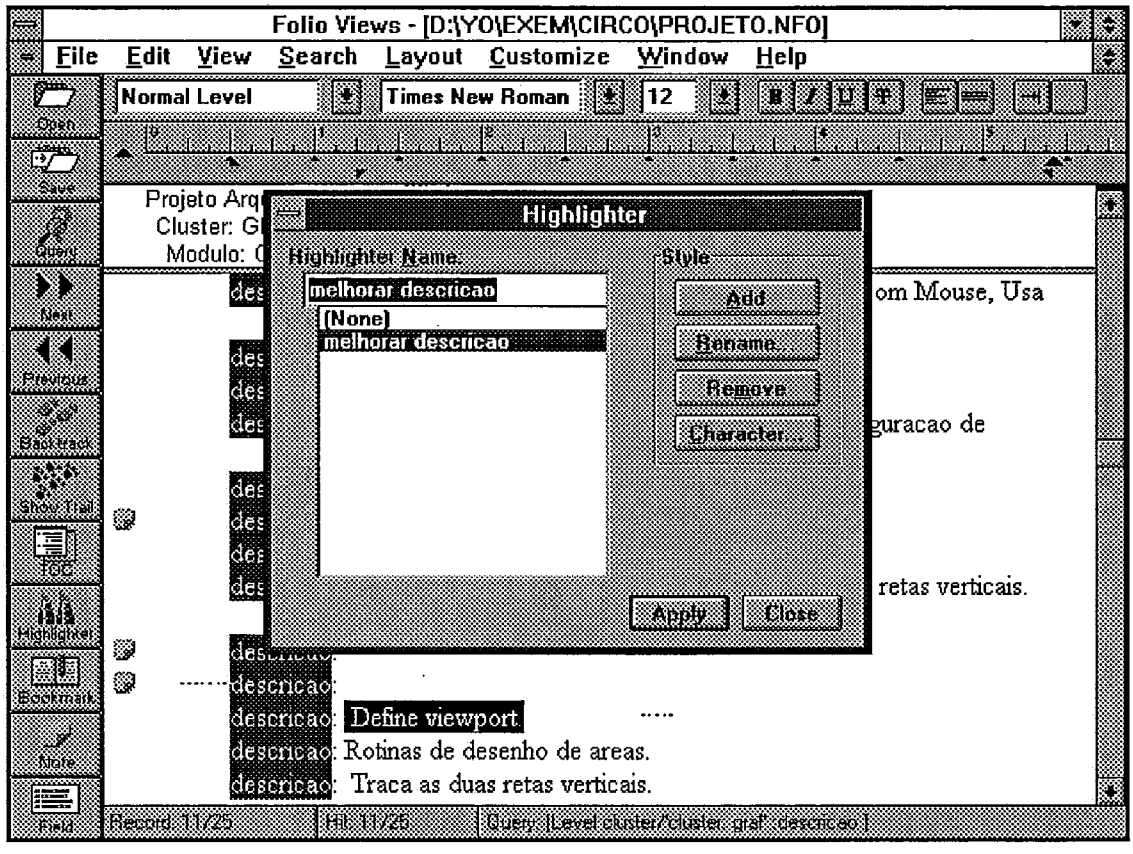


Figura 6.11: Selecionando um tipo de marcação.

O FolioViews permite ao usuário visualizar a estrutura da Infobase, através da TOC ("Table Of Contents"). A TOC é uma janela separada que mostra o conteúdo da Infobase conforme os níveis em que está organizada. Para acessá-la basta acionar o botão **TOC** situado na barra de botões. Na TOC, é possível ajustar o nível de abstração desejado, podendo visualizar apenas até um determinado nível. A estrutura da Infobase de Projeto possui cinco níveis: *sistema, cluster, módulo, interface* e *usa*. A figura 6.12 apresenta uma visão abstrata do sistema CIRCO, mostrando até o nível *módulo*, e ocultando os níveis inferiores, *interface* e *usa*.

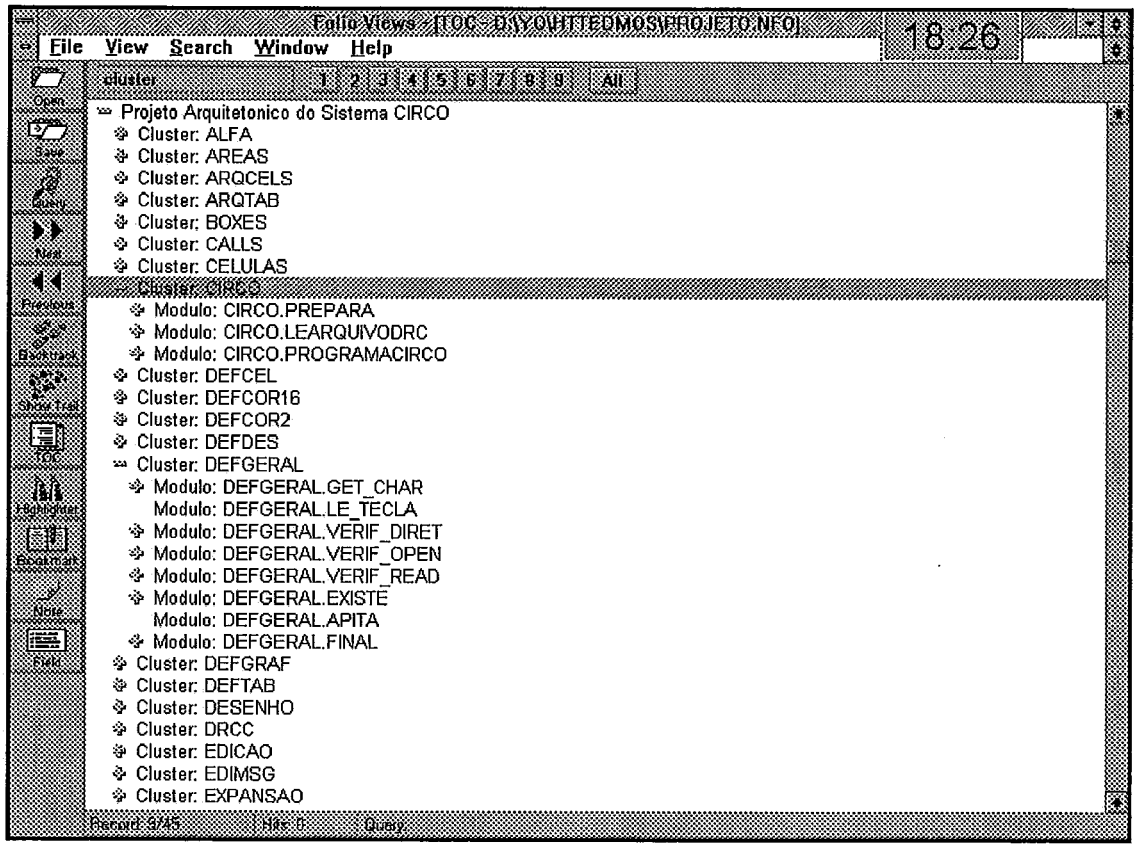


Figura 6.12: Abstração usando a TOC.

Nem todas as características e facilidades providas pelo FolioViews foram comentadas, pois uma descrição mais detalhada desta ferramenta foge ao escopo deste trabalho. Para maiores esclarecimentos aconselha-se a referência aos manuais que acompanham software [Foli93].

6.3. A notação itIBIS

A notação itIBIS, que significa "indented text Issue-Based Information System", foi desenvolvida para permitir o uso independente de hardware ou software, do modelo IBIS. A representação por endentação da estrutura hierárquica deste modelo, pode ser implementada em qualquer editor de textos [YaCo90]. Esta facilidade de implementação foi o principal motivo da escolha desta notação.

Na itIBIS usa-se os códigos I, P, AS e AO para significar respectivamente os elementos Questão ("issue"), Posição, Argumento de Suporte e Argumento de Objeção. Os símbolos "?", "*" e "-" são utilizados para significar respectivamente, pendência, aprovação/resolução e rejeição. Para identificar o proponente de uma

questão, posição ou argumento, extendemos a itIBIS, precedendo cada um destes elementos com o nome do seu proponente. Um exemplo de utilização da itIBIS pode ser visto na figura 6.13.

```
* I (Maria): Qual a notação de argumentação a ser utilizada?
        - P (ShHa94): DRL
                AO (Mário): mais complicada de usar
                AO (Mário): difícil implementação
        ? P (ShHa94): QOC
                AO (Mário): pouco conhecida
        * P (CoBe87): IBIS
                AS (YaCo90): fácil de implementar
                AS (ReEl91): simplicidade de uso
```

Figura 6.13: Exemplo de utilização da itIBIS.

A implementação da itIBIS no ambiente do FolioViews foi extremamente simples. Em paralelo à Infobase PROJETO, utiliza-se uma outra Infobase denominada DISCUTE, onde deverá ocorrer a discussão. A Infobase DISCUTE é criada a partir dos arquivos DISCUTE.DEF e DISCUTE.FFF, gerados pela ferramenta ExEM. Para implementar a itIBIS, foram definidos no arquivo DISCUTE.DEF três níveis: Questão, Posição e Argumento. Cada um destes níveis tem um estilo característico, que permite a visualização da sua posição na hierarquia. A figura 6.13 mostra o arquivo DISCUTE.DEF onde está a definição destes níveis.

```
<UG:equipe,Owner>
<MS:equipe,carlo,manoel,giga>
<UR:carlo,carlo>
<UR:manoel,manoel>
<UR:giga,giga>
<LN:sistema,cluster,modulo,questao,posicao,argumento>
<LE:"Normal Level","Normal Level",IN:1,0,0>
<LE:sistema,sistema,JU:CN,BP:0.5,BD+,FT:Helv,DV,PT:18>
<LE:cluster,cluster,BP:0.25,IN:0.25,0,-0.249306,BD+,PT:14>
<LE:modulo,modulo,BP:0.125,IN:0.5,0,-0.247917,BD+>
<LE:questao,questao,BP:0.125,IN:1.20,0,-0.6,BD+,IT+>
<LE:posicao,posicao,BP:0.125,IN:1.45,0,-0.6,PT:10>
<LE:argumento,argumento,BP:0.125,IN:3.00,0.25,-1.50,IT+,PT:10>
<CL:Bitmap>
<OD:editor,Bitmap,"C:\CIRCO\editor.BMP">
<OD:projeto,Bitmap,"C:\CIRCO\projeto.BMP">
<ST:Program,LK,FC:128,0,0>
<ST:Query,LK,FC:128,0,128>
```

Figura 6.14: Arquivo DISCUTE.DEF gerado para o sistema CIRCO.

Para iniciar a discussão sobre a complementação do documento de Projeto, uma primeira Questão surge, a partir da análise dos códigos fonte, quando um módulo não possui uma descrição no formato esperado: "Qual a descrição deste módulo?". A ferramenta ExEM é capaz de encontrar nos códigos fonte, comentários que aparecem nas linhas correspondentes a cada módulo. Estes comentários são a primeira Posição para solucionar a Questão levantada. Como um Argumento a favor desta Posição, sabe-se que os comentários relativos a um módulo podem ajudar bastante na sua descrição. Por outro lado, um Argumento contra esta mesma Posição é o fato dos comentários estarem fora de uma apresentação adequada. Este princípio de discussão é lançado pela própria ferramenta ExEM, gerando no arquivo DISCUTE.FFF, os registros necessários a criação da Infobase DISCUTE. A figura 6.15 mostra um trecho do arquivo gerado para o sistema CIRCO.

```
<DI:"DISCUTE.DEF">
<RD:sistema>Dicussão sobre o Projeto Arquitetonico
   do Sistema CIRCO

<RD:Cluster>Cluster: AREAS
  <PL:program,"d:\bp\bin\bp C:\CIRCO\AREAS.pas">
    <OB:editor,Bitmap><EL>
  <QL:Query,"Cluster AREAS","C:\CIRCO\projeto.NFO">
    <OB:projeto,Bitmap><EL>
<RD:questao> I (ExEM): Qual a descricao deste cluster?
<RD:posicao> ? P (ExEM): Esta pode ser a descricao do cluster:
{    Funcoes gerais de manipulacao de areas.                    }
<RD:argumento>AS (ExEM): Comentarios encontrados no aquivo fonte,
    no trecho correspondente a este cluster.
<RD:argumento>AO (ExEM): Texto mal escrito e improprio para ser usado
    na descricao do cluster.

<RD:Modulo>Modulo: AREAS.MUDA_VIS
  <PL:program,"d:\bp\bin\bp C:\CIRCO\AREAS.pas">
    <OB:editor,Bitmap><EL>
  <QL:Query,"Modulo AREAS.MUDA VIS","C:\CIRCO\projeto.NFO">
    <OB:projeto,Bitmap><EL>
<RD:questao> I (ExEM): Qual a descricao deste modulo?
<RD:posicao> ? P (ExEM): Esta pode ser a descricao do modulo:
  {    Modifica camadas da visibilidade.                        }
  area: RETANGULO;    { Area de redesenho }
        { se Mouse nao Instalado ou Instalado,
          mas chamada via teclado, executa trecho }
    { Le camadas, atualiza e mostra camadas visiveis. }
    begin        { Mouse Instalado -> visibpint contem cam. visiveis }
<RD:argumento>AS (ExEM): Comentarios encontrados no aquivo fonte,
        no trecho correspondente a este modulo.
<RD:argumento>AO (ExEM): Texto mal escrito e improprio para ser usado
        na descricao do modulo.

<RD:Modulo>Modulo: AREAS.APAGA_CEL
  <PL:program,"d:\bp\bin\bp C:\CIRCO\AREAS.pas">
    <OB:editor,Bitmap><EL>
  <QL:Query,"Modulo AREAS.APAGA CEL","C:\CIRCO\projeto.NFO">
    <OB:projeto,Bitmap><EL>
<RD:questao> I (ExEM): Qual a descricao deste modulo?
<RD:posicao> ? P (ExEM): Esta pode ser a descricao do modulo:
  {    Apaga a celula atual.                                    }
  {    Foi originado pela divisao do procedimento Zera_Cel      }
  {    Adaptacao para ser utilizada com menu do mouse           }
<RD:argumento>AS (ExEM): Comentarios encontrados no aquivo fonte,
        no trecho correspondente a este modulo.
<RD:argumento>AO (ExEM): Texto mal escrito e improprio para ser usado
        na descricao do modulo.
```

Figura 6.15: Trecho do arquivo DISCUTE.FFF gerado para o sistema CIRCO.

Da mesma forma como é feito para o arquivo PROJETO.FFF, ao solicitar a "abertura" do arquivo DISCUTE.FFF utilizando o tipo "Folio Flat File", a conversão é disparada automaticamente. Ao final da conversão, o usuário passa a manipular um arquivo chamado DISCUTE.NFO que corresponde à Infobase onde se dará a discussão sobre

o documento de Projeto. A figura 6.16 mostra, na forma textual, como fica um trecho da Infobase gerada a partir dos arquivos DISCUTE.DEF e DISCUTE.FFF.

Discussão sobre o Projeto Arquitetônico do sistema CIRCO

Cluster: AREAS 

I (ExEM): Qual a descrição deste cluster?

? P (ExEM): Esta pode ser a descrição do cluster:
{ Funcoes gerais de manipulacao de areas. }

AS (ExEM): Comentários encontrados no aquivo fonte, no trecho correspondente a este cluster.

AO (ExEM): Texto mal escrito e impróprio para ser usado na descrição do cluster.

Módulo: AREAS.MUDA_VIS 

I (ExEM): Qual a descrição deste módulo?

? P (ExEM): Esta pode ser a descrição do Módulo:
{ Modifica camadas da visibilidade. }
area: RETANGULO; { Area de redesenho }
{ se Mouse nao Instalado ou Instalado,
mas chamada via teclado, executa trecho }
{ Le camadas, atualiza e mostra camadas visiveis.}
begin { Mouse Instalado -> visibpint contem cam. visiveis }

AS (ExEM): Comentários encontrados no aquivo fonte, no trecho correspondente a este módulo.

AO (ExEM): Texto mal escrito e impróprio para ser usado na descrição do módulo.

Módulo: AREAS.APAGA_CEL 

I (ExEM): Qual a descrição deste módulo?

? P (ExEM): Esta pode ser a descrição do Módulo:
{ Apaga a celula atual. }
{ Foi originado pela divisao do procedimento Zera_Cel }
{ Adaptacao para ser utilizada com menu do mouse }

AS (ExEM): Comentários encontrados no aquivo fonte, no trecho correspondente a este módulo.

AO (ExEM): Texto mal escrito e impróprio para ser usado na descrição do módulo.

Figura 6.16: Trecho extraído da Infobase DISCUTE.NFO gerada para o sistema CIRCO.

Os ícones que aparecem na figura 6.16 são botões que disparam os seguintes links, previamente preparados pela ferramenta ExEM:

- *query link* com uma consulta restrita a Infobase PROJETO, para encontrar o módulo correspondente a discussão em que o usuário está posicionado;

- *program link* que dispara o editor de Turbo Pascal da Borland com o fonte correspondente ao módulo ou cluster em que o usuário está posicionado.

Este princípio de discussão deverá incentivar a equipe de manutenção do sistema, a iniciar a etapa cooperativa da Recuperação do Projeto Arquitetônico. A partir deste momento a equipe deverá utilizar a itIBIS, dentro do ambiente do FolioViews, para dar prosseguimento à discussão. A principal preocupação que os membros da equipe devem ter, é com a utilização correta dos níveis Questão, Posição e Argumento, ao acrescentar os elementos IBIS de mesmo nome. A figura 6.17 mostra como associar o nível Posição a um registro em inclusão.

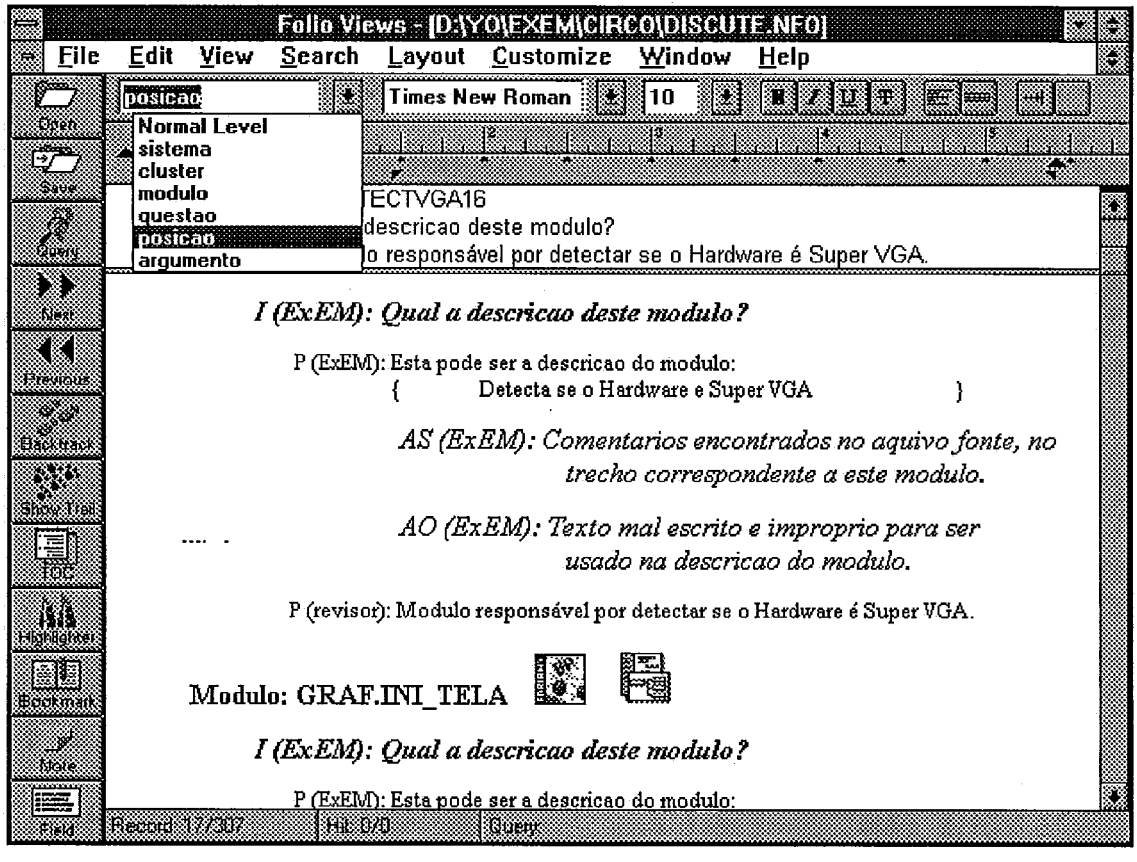


Figura 6.17: Inserindo uma nova Posição na Infobase Discussão.

A facilidade de consultas do FolioViews traz muitas vantagens, porém impõe uma limitação na implementação da itIBIS. Os símbolos ?, * e - não são reconhecidos como

palavras a consultar. Tornando sem efeito, por exemplo, uma busca sobre as questões que já foram resolvidas. Por isso, fomos obrigados a substituir estes símbolos pelos caracteres alfabéticos "d", "s" ou "r". respectivamente. Os exemplos abaixo ilustram a utilização desta nova notação:

- ***questão solucionada***
  s I (Maria): O que se pode comentar sobre a interface deste módulo?
- ***posição em dúvida***
  d P (Carlos): O módulo espera um inteiro positivo menor do que 13.
- ***posição rejeitada***
  r P (Eduardo): O módulo espera um inteiro positivo.
- ***posição selecionada***
  s P (Zé): O módulo espera um inteiro positivo menor que 13 e diferente de zero.

Apesar desta pequena limitação, o mecanismo de consultas do FolioViews é uma de suas características mais interessantes. Durante a Discussão de Recuperação do Projeto, este mecanismo possui um papel importante, permitindo que os usuários localizem-se rapidamente na Infobase. Entre consultas úteis à Infobase de Discussão podemos citar:

- ***Quais as propostas pendentes de determinado cluster?***
  [Level cluster/cluster X: 'd P']
- ***Quais as questões solucionadas do sistema?***
  (s I)
- ***Quais os argumentos contra as posições de Ana?***
  [Level posição:'P (Ana)'][Level argumento: AO]

A escolha das ferramentas ExEM e FolioViews, e da notação itIBIS, para implementar o protótipo ARCoPAS, baseou-se nos requisitos levantados no capitulo 5. Entretanto, para a verificação do preenchimento destes requisitos foi preciso realizarmos um experimento, onde pudemos constatar as falhas e virtudes deste protótipo.

Capítulo 7
Observações e Resultados

Após a implementação do protótipo ARCoPAS, efetuamos uma análise do mesmo, comparando-o com o que se considera um ambiente ideal para a Recuperação de Projeto. Iniciamos também, um experimento, onde foi possível identificar as virtudes e deficiências apresentadas pelo protótipo. A próxima versão do ARCoPAS incorporará melhorias que deverão eliminar as deficiências encontradas.

7.1. O Ambiente Ideal

Revendo as características desejáveis identificadas por Fletton e Munro [FlMu88] para um ambiente de Recuperação de Projeto, podemos observar que o ARCoPAS satisfaz a muitas delas:

- ✓ **Incremental:** A ferramenta FolioViews e a notação itIBIS encarregam-se de tornar o ARCoPAS incremental, permitindo que o conhecimento dos especialistas seja estruturada e assincronamente capturado. A estruturação da informação mantém a coerência durante o processo de Recuperação, evitando que os especialistas se dispersem. O assincronismo dispensa a necessidade de reuniões freqüentes, minimizando a concorrência com as atividades mais urgentes dos especialistas.

- ✓ **Atualização Casual:** O ARCoPAS encontra-se no próprio ambiente de trabalho dos responsáveis pela tarefa de Recuperação de Projeto, possibilitando-lhes um fácil acesso. Mas além disso, o ARCoPAS proporciona o desenvolvimento de uma discussão em grupo, tornando o processo de obtenção da informação quase natural.

- ✓ **Garantia de Qualidade:** O ARCoPAS não provê esta característica, porém, a utilização de uma ferramenta de apoio à discussão sobre o Projeto Arquitetônico (FolioViews + itIBIS), é um passo importante na direção da melhoria da qualidade do mesmo.

- ✓ **Trabalho Cooperativo:** O ARCoPAS pode ser encarado como uma tecnologia de suporte ao trabalho cooperativo. Segundo a classificação de groupwares apresentada no capítulo 4, o ARCoPAS encaixa-se em dois tipos de groupware,

sendo ao mesmo tempo, uma ferramenta de apoio à discussão (FolioViews + itIBIS), e uma ferramenta de co-autoria (FolioViews).

✓ **Gerência de Configuração:** A facilidade de utilizar cópias de uma Infobase ("shadow files"), permite que se trabalhe em novas versões destes documentos. Porém, ao contrário do que se poderia esperar, estas cópias não podem ser incorporadas às originais. Na verdade, os "shadow files" são máscaras, que possuem o objetivo de proteger a Infobase original de alterações indesejáveis. Assim, por enquanto, o ARCoPAS deixa a tarefa de gerência de configuração por conta dos seus usuários.

✓ **Integração ao Código Fonte:** Os *program links* do Folio Views, permitem que os usuários do ARCoPAS tenham fácil acesso ao código fonte relacionado a documentação de Projeto. Uma vez dentro da documentação de Projeto, basta acionar um botão, que o ARCoPAS transporta o usuário para o ambiente de programação, onde este pode visualizar imediatamente, o código fonte correspondente.

✓ **Integração com ferramentas automáticas:** O ARCoPAS é um ambiente formado pela integração de duas ferramentas automáticas (ExEM e FolioViews), o que lhe confere naturalmente esta característica. Além disso, o Folio Views permite o intercâmbio de arquivos com diversas outras ferramentas, permitindo a importação de vários documentos e a incorporação destes ao documento de Projeto.

✓ **Ocultamento de Informação:** Esta é outra característica provida pelo ARCoPAS. A possibilidade de definição de níveis hierárquicos no Folio Views, permite que os usuários visualizem os documentos de Projeto e de Discussão, com maior ou menor abstração.

7.2. Experimento

Para avaliar o ARCoPAS estamos aplicando-o a sistemas reais, que preencham alguns requisitos básicos. Em primeiro lugar, o sistema deve encontrar-se em fase de manutenção, isto é, deve haver pelo menos uma versão concluída e em operação. Outro requisito importante é a existência de uma equipe composta de duas ou mais pessoas, entre analistas e programadores, mantendo o sistema. Além destes dois

requisitos principais, é necessário que o sistema possua um tamanho razoável, i.e., da ordem de milhares de linhas de código [SeWaCh93].

Para realizar um primeiro experimento escolhemos o Sistema TEDMOS, pois este sistema preenche todos os requisitos acima citados. O TEDMOS é uma ferramenta CAD para ensino e projeto de Circuitos Integrados [AnOl92]. Considerado como um sistema de médio porte, o TEDMOS possui mais de 57.000 linhas de código escrito em Turbo Pascal para funcionar em ambiente PC/DOS. Este sistema foi desenvolvido pelo Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro, e encontra-se atualmente na versão 5.0 a caminho da 5.1. Alguns dos exemplos mostrados neste trabalho são resultado da utilização de fontes de um subsistema do TEDMOS chamado CIRCO. O subsistema CIRCO corresponde a aproximadamente metade do tamanho total do TEDMOS.

A Recuperação do sistema TEDMOS ainda encontra-se em andamento. Durante a sua fase inicial, pudemos observar o que se passou entre dois membros da equipe: Maria e Manoel. Maria é uma analista que participa temporariamente, da equipe do TEDMOS, e Manoel é o gerente desta equipe há alguns anos. Empenhada em conhecer o sistema, Maria utilizou o ARCoPAS para facilitar o seu estudo, enquanto dava início à recuperação do Projeto Arquitetônico do mesmo. Tanto para Maria quanto para Manoel, o primeiro contato com o ARCoPAS não foi precedido por um treinamento formal nas ferramentas que o integram. Após uma explicação rápida, ambos familiarizaram-se com as funções básicas do ambiente.

Observamos que entre as facilidades mais usadas do ARCoPAS, estão o acesso imediato ao código fonte, a navegação pela estrutura modular e o mecanismo de consulta. O botão que transporta o usuário para o programa fonte, foi muito utilizado, confirmando a importância da integração entre os documentos do sistema. A navegação pela estrutura modular foi útil para orientar os estudos de Maria. No entanto, durante um dos passeios pela estrutura modular do sistema, Maria acusou problemas de desorientação. Estes problemas logo foram sanados pela facilidade "show trail" do FolioViews. Ao acionar o botão "show trail", situado na barra de botões, Maria pôde ver todo o caminho que havia percorrido na Infobase até então, retornando rapidamente ao ponto desejado (veja a figura 7.1).

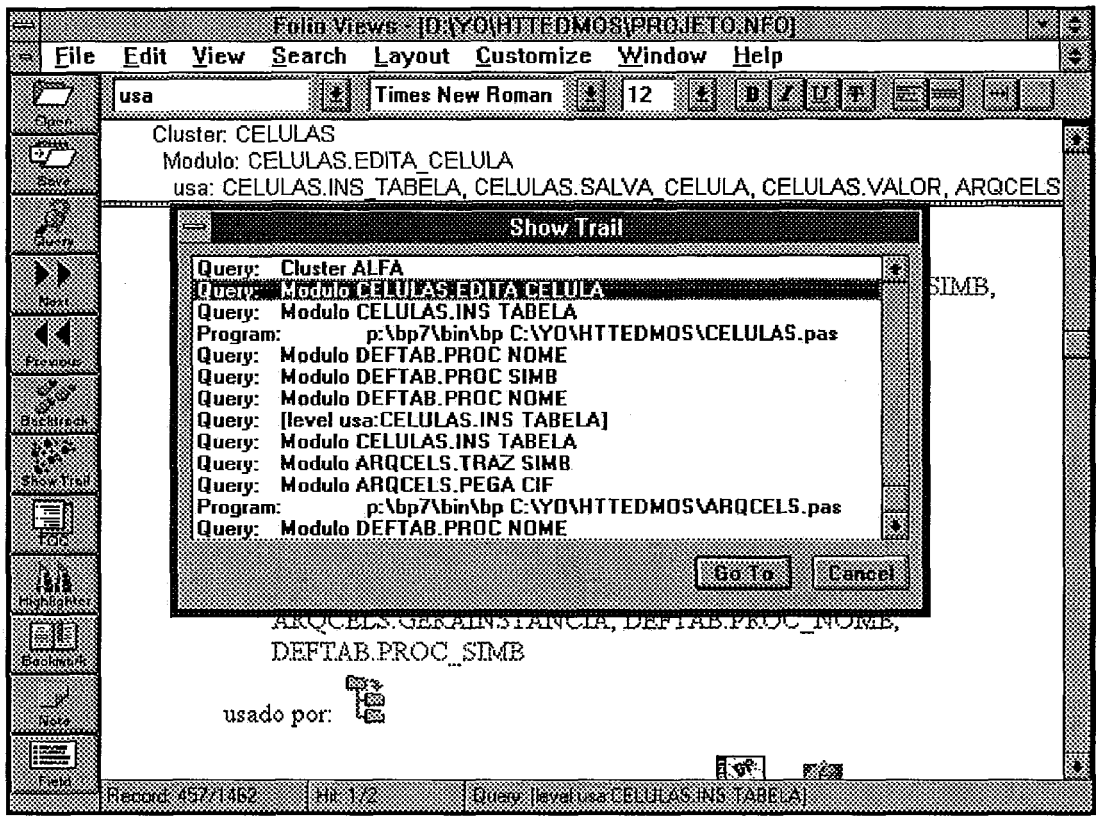


Figura 7.1: Visualizando o caminho percorrido.

O grande número de clusters do sistema TEDMOS (totalizando 128) deixou Maria em dúvida quanto a ecolha do cluster por onde iniciar seu estudo. Sabendo de antemão, que o CIRCO era um dos principais clusters do sistema, Maria o escolheu. Esta desorientação inicial, no entanto, mostra que a clusterização existente no sistema TEDMOS, não foi suficiente para gerar sua árvore conceitual automaticamente. Maria precisaria encontrar níveis intermediários de clusterização, entre o sistema e os clusters, de maneira a facilitar a sua compreensão do sistema e a escolha de um ponto de partida. Caso Maria houvesse solicitado a ajuda de Manoel, ele poderia criar os níveis intermediários, agrupando os clusters em "super clusters", segundo a sua experiência no sistema.

Maria já havia iniciado seu estudo, colocando algumas posições sobre a descrição dos módulos do cluster CIRCO, quando Manoel acessou a Infobase da Discussão. Com o intuito de ajudá-la, Manoel utilizou o mecanismo de consulta do FolioViews, para encontrar na Infobase, as posições em que Maria possuía dúvida. No entanto, a utilização deste mecanismo não é trivial, dificultando a ação de Manoel. Após uma breve consultoria a um usuário do FolioViews, Manoel pôde localizar as dúvidas de Maria, como mostra a figura 7.2.

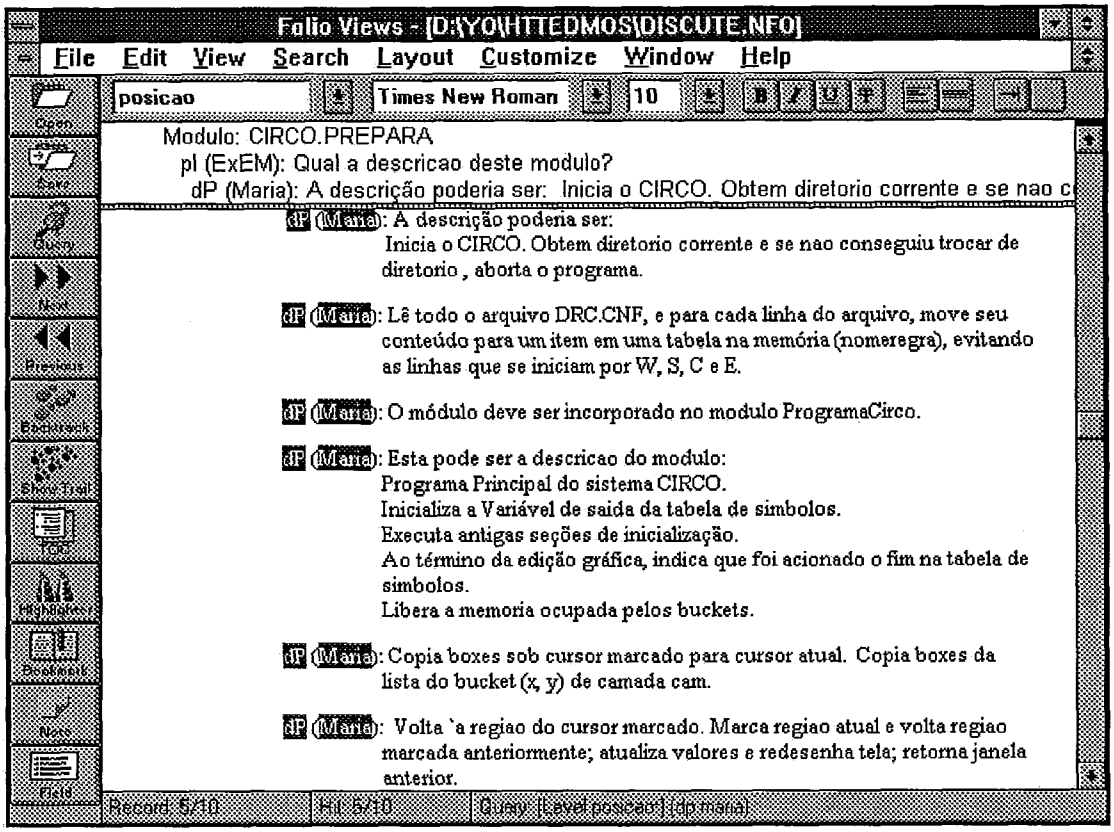


Figura 7.2: Consultando as dúvidas de Maria.

Ao tentar sanar as tais dúvidas, Manoel precisou consultar arquivos de dados do sistema. Como a ferramenta ExEM só considerou os arquivos fonte do sistema, os arquivos de dados não constam nas Infobases como módulos do sistema. Assim, para alcançá-los foi preciso utilizar o botão de acesso ao ambiente de programação, pertencente a um módulo de programa fonte. Uma vez no Turbo Pascal, foi possível abrir e visualizar o arquivo de dados correspondente. Após estar seguro de suas posições, Manoel as incluiu na Infobase de Discussão, como mostram as figuras 7.3 e 7.4.

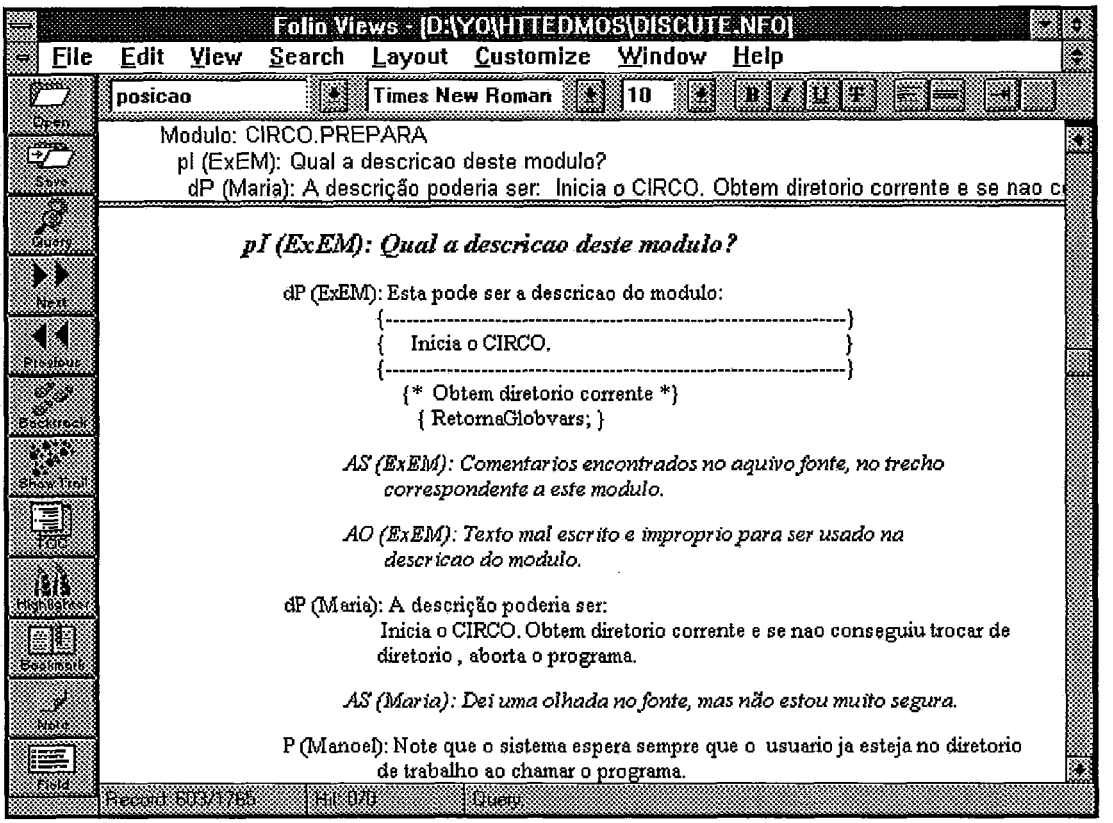


Figura 7.3: Complementando a descrição do módulo PREPARA.

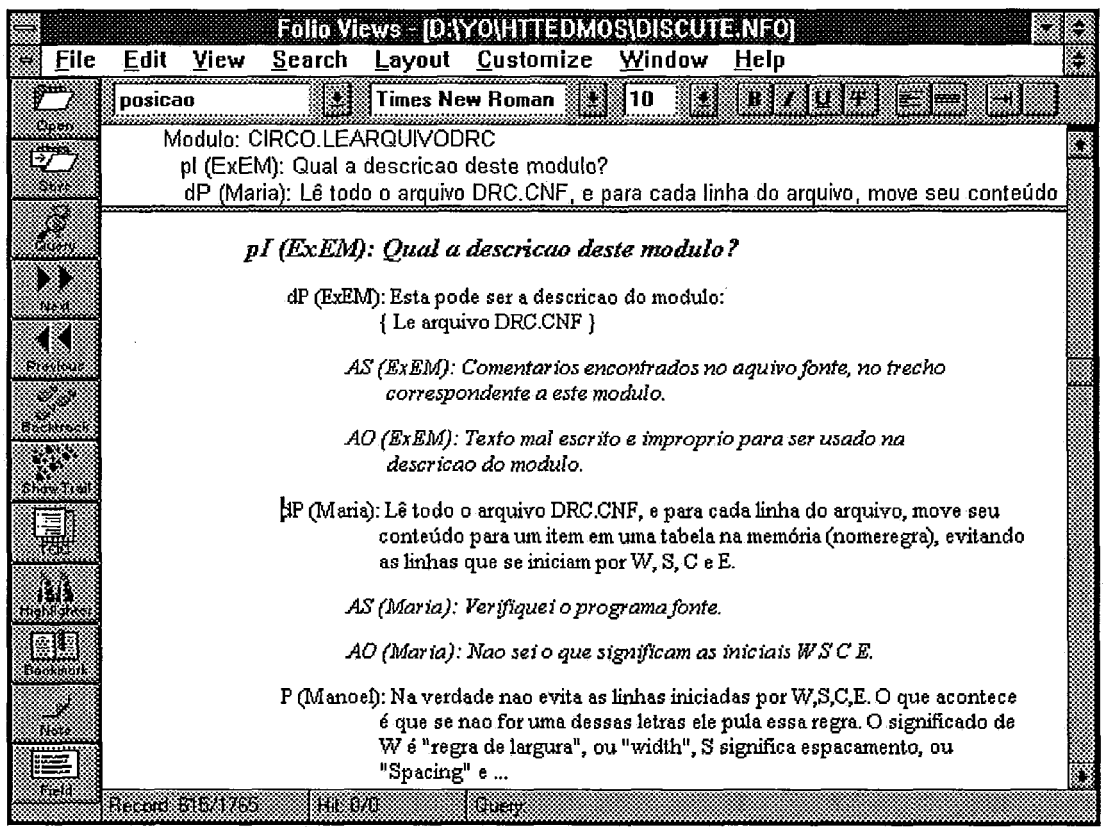


Figura 7.4: Complementando a descrição do módulo LEARQUIVODRC.

7.3. Melhorias Futuras

Futuramente, pretendemos tornar o protótipo ARCoPAS um ambiente mais completo e flexível, suprindo as deficiências apresentadas. Uma melhoria prevista para breve, é o tratamento dos arquivos de dados. Além dos arquivos fonte (.PAS), a ferramenta ExEM passará a identificar como módulos, os arquivos de dados usados pelo sistema TEDMOS (.CNF, .MSG, .HLP e .USU). A prioridade desta mudança deve-se à requisição feita pelos membros da equipe TEDMOS.

Outra importante melhoria refere-se à extração da árvore conceitual. Como foi observado, a ferramenta ExEM não consegue extrair mais do que três níveis de abstração. Precisamos investigar a possibilidade de encontrar dentro dos programas fonte, alguma pista que determine a existência de níveis intermediários de abstração, com o objetivo de gerar uma árvore mais "alta" e consequentemente, mais representativa.

Como foi comentado na sessão anterior, falta pouco para que o ARCoPAS se torne o ambiente ideal. Em uma próxima versão o ARCoPAS pretende prover a garantia de qualidade e o controle de versão dos documentos que gera. Para garantir a qualidade da árvore conceitual gerada, a ferramenta ExEM deverá sugerir mudanças e criar níveis na mesma, baseando-se em algorítmos de clusterização por similaridade [ScPl89]. O ARCoPAS deverá prover meios de fomentar a discussão em andamento, instigando a participação dos membros do grupo, aumentando a qualidade não só do objeto em discussão, como da própria discussão.

Para prover uma completa gerência de configuração, talvez seja necessário mudar a ferramenta de hipertexto (FolioViews). Porém, mesmo com a mesma ferramenta, podemos garantir a correspondência entre as versões da documentação de Projeto e dos códigos fonte, sendo necessário fazer apenas algumas adaptações no ARCoPAS.

Notamos também, a necessidade da definição de papéis para a tarefa de recuperação de Projeto. O mecanismo de segurança do FolioViews permite a identificação dos usuários, a formação de grupos e a restrição dos direitos de cada um. Esta facilidade possibilita uma fácil implementação de papéis. Entretanto, uma definição prematura, poderia prejudicar as pesquisas com relação ao aspecto social do grupo em estudo. A partir de experiências com a utilização do ARCoPAS, esperamos obter conhecimento suficiente para uma definição adequada dos papéis.

O tratamento de uma única linguagem de programação (Pascal), é uma restrição que pretendemos eliminar. É nossa intensão, reprojetar a ferramenta ExEM, utilizando para

isso o Projeto Arquitetônico recuperado da mesma (veja o Apêndice). O próprio ARCoPAS será usado para o reprojeto da ExEM, mudando apenas o objetivo da discussão. Neste reprojeto, pretendemos prever o tratamento de outras linguagens tais como C e COBOL. Entretanto, nada impede que um outro protótipo da ExEM, dedicado a outra linguagem de programação, seja implementado a partir da reutilização do seu Projeto Arquitetônico. Enquanto a ExEM genérica não fica pronta, as equipes de manutenção interessadas poderão implementar por exemplo, uma $ExEM_C$ ou uma $ExEM_{cobol}$, adaptando o ARCoPAS às suas necessidades.

A primeira versão da ferramenta ExEM está pouco otimizada. O fato de lidar com muitos arquivos de dados torna a ferramenta extremamente lenta. Uma vez que a ExEM não precisa ser constantemente executada, não há muita urgência em efetuar esta melhoria. Entretanto, esta otimização está prevista para o Reprojeto da mesma.

Para que uma equipe aceite utilizar uma nova ferramenta, é preciso contar com o apoio da gerência da equipe, e prover treinamento e suporte técnico a esta ferramenta [YaCo90]. Em nosso primeiro experimento, contamos com o apoio gerencial, porém não oferecemos um treinamento adequado, contando apenas com a facilidade de uso das ferramentas. Em futuras experiências, pretendemos planejar mais elaboradamente, a introdução do ARCoPAS.

Capítulo 8
Conclusão

Este trabalho propõe a utilização de um ambiente cooperativo para a Recuperação do Projeto Arquitetônico de Sistemas. Os problemas da fase de manutenção de software foram os motivadores para o desenvolvimento deste trabalho. Como abordagem para atacar estes problemas, aliamos à Engenharia Reversa, as tecnologias de Hipertexto e CSCW, idealizando um ambiente para a Recuperação de Projeto. Desenvolvemos um protótipo baseado no ambiente proposto, o ARCoPAS, que integra duas ferramentas: ExEM e FolioViews. Para avaliar a usabilidade deste ambiente, iniciamos um experimento, aplicando o ARCoPAS a um sistema real.

A introdução da característica cooperativa a uma ferramenta de apoio à Recuperação de Projeto traz uma importante contribuição para a Engenharia Reversa, já que poucas ferramentas a incluem. Primeiramente, viabilizamos a participação humana, possibilitando a captura do conhecimento dos especialistas (equipe de manutenção), e enriquecendo com isto, o produto recuperado. Em segundo lugar, a distribuição da responsabilidade desta tarefa entre os membros da equipe de manutenção, torna-a mais interessante e menos desgastante. Por último, a desobrigação de uma cooperação síncrona, pouco prejudica as demais e urgentes tarefas desta equipe.

O ARCoPAS preenche seis dos oito requisitos desejáveis, levantados por Fletton e Munro [FlMu88]. A *atualização incremental* e *casual*, a *integração com o código fonte* e com outras *ferramentas automáticas*, e o *ocultamento da informação* são providos pelas ferramentas Folio Views e ExEM. O *trabalho cooperativo* é suportado pela implementação da notação itIBIS sobre o Folio Views. O ARCoPAS não provê a *garantia de qualidade* e a *gerência de configuração*, entretanto o planejamento de melhorias já prevê o preenchimento destes requisitos. Estão previstas também, melhorias para a ferramenta ExEM, tais como a análise de código escrito em outras linguagens de programação e a geração de uma árvore conceitual mais alta.

Os ambientes de apoio à Engenharia Reversa devem buscar a integração com ambientes de apoio à Engenharia Progressiva. Esta integração facilita a introdução de novas metodologias nos ambientes de manutenção e desenvolvimento de software. O protótipo ARCoPAS introduz uma metodologia de Reprojeto Cooperativo à equipe de manutenção de um sistema, propondo a utilização da notação itIBIS para capturar as decisões de Reprojeto. Ao utilizar o ARCoPAS para recuperar o Projeto de um sistema, os membros da equipe familiarizam-se com a notação itIBIS. Durante as discussões sobre o Projeto, geralmente surgem questões que propõem mudanças no

mesmo, e que acabam gerando decisões de Reprojeto. Assim sendo, o Reprojeto Cooperativo de Sistemas torna-se uma consequência natural da utilização do ARCoPAS.

A observação do uso do protótipo ARCoPAS, pode ser utilizada como parte de um estudo etnográfico sobre as equipes de manutenção de software e a tarefa de recuperação de Projeto. Os resultados obtidos com este estudo, poderão contribuir para o enriquecimento das pesquisas na área de CSCW. Através da realização de alguns experimentos, esperamos recolher material suficiente para a construção de um ambiente definitivo.

Apêndice
Projeto Arquitetônico da Ferramenta ExEM

Cluster: ExEM

D:\ExEM 2-8-1994 0:45

descrição: O sistema ExEM é um extrator da estrutura modular de um sistema escrito em Turbo Pascal. Foi desenvolvido por Maria Cláudia Reis Cavalcanti, no período de maio a julho de 1994.

Módulo: ExEM

descrição: Este é o programa principal do sistema ExEM, que chama os módulos responsáveis pela identificação dos módulos e relacionamentos do sistema em análise (IDMODULO), e chama os módulos responsáveis pela geração da estrutura modular na notação HTDN (GERAHTDN).

usa: GERAHTDN.GERADEFS, GERAHTDN.GERAFFFS, IDMODULO.DESDOBUNI, IDMODULO.GERAARQMUS, IDMODULO.GERAARQUNI, IDMODULO.IDENTMODULOPROG, IDMODULO.IDENTMODULOSUNIT, IDMODULO.TIRACOM, TEMTUDO.BUSCASTRARQ, TEMTUDO.TRUNCADIR

Cluster: IDMODULO

D:\ExEM 2-8-1994 9:53

descrição: Identifica os módulos e suas relações ao mesmo tempo em que extrai suas descrições e interfaces.

Módulo: IDMODULO.GRAVACOM

descrição: Grava o comentário em CONTEUDOLIN e a posição do mesmo em CONTALIN, no arquivo .COM.

interface: (conteudolin:string; contalin:integer)

usa: TEMTUDO.BRANCOS

Módulo: IDMODULO.TIRACOM

descrição: Cria o arquivo .PSC que é uma cópia do .PAS sem comentários, gerando também o .COM com os comentários.

interface: (arquivo: string)

usa: IDMODULO.GRAVACOM, TEMTUDO.POSLIVRE

Módulo: IDMODULO.GERAARQUNI

descrição: Gera o arquivo com a lista de units usadas por uma unit (.UNI). Este arquivo será lido mais tarde para que cada unit da lista seja desdobrada em seus módulos públicos internos.

interface: (limite: string)

usa: TEMTUDO.BRANCOS, TEMTUDO.BUSCASTRARQ

Módulo: IDMODULO.ISOLANOME

descrição: Obtém o nome do identificador logo após a palavra em PRECEDENTE que ocorre na LINHA, e retorna este nome.

interface: (precedente,linha:string)
função do tipo string

usa: TEMTUDO.POSLIVRE

Módulo: IDMODULO.EXTRAIINTERF

descrição: Extrai o texto relativo aos parâmetros passados para o módulo e grava no arquivo de interfaces .INT.

interface: (var pontarq:integer; linha:string)

usa: TEMTUDO.BRANCOS, TEMTUDO.LELINHA

Módulo: IDMODULO.ACHAMODULO

descrição: Varre as linhas do arquivo .Psc da posição em que se encontra (pontArq) até encontrar uma nova definição de procedure ou function. Caso encontre, retorna True.

interface: (var pontarq:integer)
função do tipo boolean

usa: IDMODULO.EXTRAIINTERF, IDMODULO.ISOLANOME, TEMTUDO.BRANCOS, TEMTUDO.LELINHA, TEMTUDO.POSLIVRE

Módulo: IDMODULO.DELIMITABLOCO

descrição: Delimita o bloco de uma procedure/function, varrendo as linhas do arquivo .PSC a partir de PONTARQ, ignorando as definições internas de outras procedures e functions.

interface: (var pontarq: integer)
função do tipo integer

usa: TEMTUDO.LELINHA, TEMTUDO.POSLIVRE

Módulo: IDMODULO.IDENTMODULOSUNIT

descrição: Identifica os módulos públicos e privados de uma Unit. Gera os arquivos .MOD : dados de cada módulo;
.UNP : lista dos módulos públicos de uma unit

usa: IDMODULO.ACHAMODULO, IDMODULO.DELIMITABLOCO, IDMODULO.EXTRAIINTERF, MANLISTA.INSERELISTAMOD, TEMTUDO.BRANCOS, TEMTUDO.BUSCASTRARQ, TEMTUDO.POSICIONAARQ, TEMTUDO.POSLIVRE, TEMTUDO.TRUNCADIR

Módulo: IDMODULO.GERAARQMUS

descrição: Varre o arquivo .MOD, e para cada módulo da unit em anáise monta-se uma lista com os módulos usados encontrados no .PSC correspondente. Terminada a lista, esta é gravada no arquivo .MUS.

usa: MANLISTA.DISPOSELISTAMOD, TEMTUDO.BRANCOS, TEMTUDO.LELINHA, TEMTUDO.POSICIONAARQ, TEMTUDO.POSLIVRE, TEMTUDO.UPCASESTR

Módulo: IDMODULO.IDENTMODULOPROG

descrição: Os programas são considerados como um módulo pertencente a uma unit que possui o seu nome, não sendo necessário analisar os módulos internos aos mesmos. A única relação que os programas possuem com os demais módulos do sistema é através da cláusula uses, que é analisada da mesma forma que nas units.

usa: TEMTUDO.BRANCOS, TEMTUDO.BUSCASTRARQ

Módulo: IDMODULO.DESDOBUNI

descrição: Cria o arquivo .UND substituindo as units usadas do arquivo .UNI por seus módulos públicos correspondentes.

Módulo: IDMODULO.MONTALISTAEXT

descrição: Monta a lista de módulos externos da unit mãe.

interface: (var externoslist:pontext)

usa: MANLISTA.DISPOSELISTAEXT, MANLISTA.INSERELISTAEXT, TEMTUDO.TRUNCADIR

Módulo: IDMODULO.GRAVAMES

descrição: Grava o registro no arquivo .MES.

interface: (var externoslist: pontext)

usa: TEMTUDO.BRANCOS

Módulo: IDMODULO.DESCOBMES

descrição: Cria o arquivo .MES com os módulos externos usados de cada módulo. Varre trecho do fonte fazendo a interseção de cada linha do fonte com a lista de módulos externos da unit mãe. Verifica se os módulos externos e as units correspondentes estão presentes na linha do programa fonte.

usa: IDMODULO.GRAVAMES, IDMODULO.MONTALISTAEXT, TEMTUDO.LELINHA, TEMTUDO.POSICIONAARQ, TEMTUDO.POSLIVRE, TEMTUDO.TRUNCADIR, TEMTUDO.UPCASESTR

Cluster: GERAHTDN

D:\ExEM 2-8-1994 0:41

descrição: Gera os arquivos .DEF e .FFF responsáveis pela interface com o Folio Views.

Módulo: GERAHTDN.OBTEMAMBPROG

descrição: Obtem do usuário qual o path para rodar o ambiente de edição e programação utilizado.

interface: função do tipo string

Módulo: GERAHTDN.ESCREVEITEM

descrição: Escreve o título de cada item, seja cluster ou módulo, colocando os ícones devidos: um ícone para linkar com o ambiente de edição de programas, e um ícone para linkar com a outra infobase.

interface: (var arq:text; infobase,item:string)

usa: TEMTUDO.TIRAUND, TEMTUDO.TRUNCADIR

Módulo: GERAHTDN.ESCREVEUSA

descrição: Escreve no arquivo PROJETO.FFF o nível usa de cada módulo.

interface: (var titulo:string;nomeunit, nomemod: string)

usa: TEMTUDO.TIRAUND, TEMTUDO.TRUNCADIR

Módulo: GERAHTDN.ESCREVEDESC

descrição: Escreve a descrição do módulo nos arquivos PROJETO.FFF e DISCUTE.FFF.

interface: (arqpascal:string;inicio,fim: integer; item: string)

Módulo: GERAHTDN.ESCREVEINTERFACE

descrição: Le o arquivo de interfaces e escreve as linhas correspondentes no arquivo .FFF.

usa: TEMTUDO.BRANCOS

Módulo: GERAHTDN.GERAFFFS

descrição: Lê os arquivos .MUS, .MES gerados anteriormente e gera um outro com a Text Design Notation do sistema com as marcações devidas para tornar-se uma Infobase do FolioViews (arquivo .FFF).

usa: GERAHTDN.ESCREVEDESC, GERAHTDN.ESCREVEINTERFACE, GERAHTDN.ESCREVEITEM, GERAHTDN.ESCREVEUSA, GERAHTDN.OBTEMAMBPROG, TEMTUDO.OBTEMDATAHORA, TEMTUDO.TIRAUND, TEMTUDO.TRUNCADIR

Módulo: GERAHTDN.GERADEFS

descrição: Gera o arquivo .DEF necessário as definições de estilo dos níveis hierárquicos gerados no arquivo .FFF.

Cluster: MANLISTA

D:\ExEM 1-8-1994 23:59

descrição: Unit para manipulação de listas ordenadas.

Módulo: MANLISTA.INSERELISTAEXT

descrição: Insere NOVO na lista de módulos externos apontada por PRIMEIRO.

interface: (var primeiro, novo: pontext)

Módulo: MANLISTA.INSERELISTAMOD

descrição: Insere NOVO na lista de módulos internos apontada por PRIMEIRO.

interface: (var primeiro, novo: pontmod)

Módulo: MANLISTA.DISPOSELISTAEXT

descrição: Libera a área de memória da lista de módulos externos.

interface: (var lista: pontext)

Módulo: MANLISTA.DISPOSELISTAMOD

descrição: Libera a área de memória da lista de módulos internos.

interface: (var lista: pontmod)

Cluster: TEMTUDO

D:\ExEM 1-8-1994 23:59

descrição: Biblioteca de procedures e functions relativas ao tratamento de arquivos, e definições globais..

Módulo: TEMTUDO.BUSCASTRARQ

descrição: Busca uma string AGULHA em um arquivo PALHEIRO, até encontrar uma string LIMITE, a partir da posição POSARQ, atualizando a variável linha e retornando o número da linha do arquivo em que encontrou a AGULHA.

interface: (agulha, limite: string; var palheiro: text; posarq: integer; var linha:string)
função do tipo integer

usa: TEMTUDO.POSICIONAARQ, TEMTUDO.POSLIVRE, TEMTUDO.UPCASESTR

Módulo: TEMTUDO.POSICIONAARQ

descrição: Posiciona o ARQUIVO texto na Posição desejada.

interface: (var arquivo: text; posição: integer)

Módulo: TEMTUDO.UPCASESTR

descrição: Transforma uma cadeia em letras maiúsculas.

interface: (linha: string)
função do tipo string

Módulo: TEMTUDO.TRUNCADIR

descrição: Trunca os brancos a direita de uma cadeia.

interface: (var cadeia:string)

Módulo: TEMTUDO.TIRAUND

descrição: Substitui o underscore de uma cadeia por branco.

interface: (var cadeia:string)

Módulo: TEMTUDO.SIMBOLO

descrição: Retorna true se a cadeia apresentada contem algum símbolo que não pode fazer parte de uma palavra ou identificador Pascal.

interface: (cadeia:string)
função do tipo boolean

Módulo: TEMTUDO.PALAVRA

descrição: Verifica se a cadeia procurada na linha constitui uma palavra - true, ou se faz parte de uma cadeia maior.

interface: (cadeia,linha:string)
função do tipo boolean

usa: TEMTUDO.SIMBOLO

Módulo: TEMTUDO.POSLIVRE

descrição: Verifica se CADEIA encontra-se na LINHA livre de CARRASCO, isto é, fora de um intervalo de abre/fecha CARRASCO (até o momento só se aplica a Pliques).

interface: (cadeia:string;carrasco:char;linha:string)
função do tipo integer

usa: TEMTUDO.PALAVRA

Módulo: TEMTUDO.BRANCOS

descrição: Retorna uma cadeia com N brancos.

interface: (numero:integer)
função do tipo string

Módulo: TEMTUDO.OBTEMDATAHORA

descrição: Obtém a data e a hora do arquivo passado por parâmetro.

interface: (arquivo:string; var data,hora: string)

Módulo: TEMTUDO.LELINHA

descrição: Lê a próxima linha do ARQ, transforma-a em maiúsculas e atualiza o contador de linhas.

interface: (var arq: text; var linha: string; var numlinha: integer)

usa: TEMTUDO.UPCASESTR

Bibliografia

[AbBa93] "Documenting Programs using a Library of Tree Structures Plans" - S. Abd-El-Hafiz & V. Basili, Proceedings of the Conference on Software Maintenance, 1993;

[Akta87] "Structured Analysis and Design of information systems" - A. Z. Aktas, Prentice Hall International Editions, 1987;

[AnOl92] "A CAD system for teaching the design of VLSI Circuits - Status and Evolution" - M. L. Anido & C. E. Oliveira, PANEL'92 - 18th Latin American Informatics Conference, Las Palmas de G.C., Espanha, 1992;

[BeCiDe89] "Reverse Engineering Methodology to Reconstruct Hierarquical Data Flow Diagrams for Software Maintenance" - P. Benedusi & A. Cimitile & U. De Carlini, Proceedings of the Conference on Software Maintenance, 1989;

[Benn91] "Automated support of software maintenance" - K. H. Bennett, Information and Software Technology, vol 33, Num 1, janeiro/fevereiro 1991;

[Bige88] "Hypertext and CASE" - J. Bigelow, IEEE Software, março/1988;

[Bigg89] "Design Recovery for Maintenance and Reuse" - T J Biggerstaff, Computer, pag 36-49, julho/1989;

[Bigg90] "Human-Oriented Conceptual Abstractions in the Re-engineering of Software" - T. Biggerstaff, IEEE Proceedings da 12a Conf. Internacional de Engenharia de Software, 1990;

[BiRi87] "Manipulating Source Code in DynamicDesign" - J. Bigelow & V. Riley, Proceedings Hypertext'87, novembro/87;

[Blom93] "Ethnographic Field Methods and Their Relation to Design" - J. Blomberg, Participatory Design: Principles and Practices", ed. Douglas Schuler & Aki Namioka, 1993;

[Booc86] "Object-Oriented Development" - G. Booch, IEEE Transactions on Software Engineering, vol 12(2), fevereiro/86;

[Booc94] "Object-Oriented Analysis and Design with applications" - G. Booch, 2nd edition, The Benjamin/Cummings Pub. Co. Inc., 1994;

[Borg91] "Contribution to Further Development of JSD" - M. Borgers, Information and Software Technology, vol 33, n. 5, junho/1991;

[Borg93] "Suporte por Computador ao Trabalho Cooperativo" - M. R. S. Borges, VI Escola Brasil-Argentina de Informática, julho/1993;

[Borl93] "Borland Pascal User's Guide" - Borland, 1993;

[Came88] "The Modelling Phase of JSD" - J. R. Cameron, Information and Software Technology, vol. 30, n. 6, julho/1988;

[CaBo94] "ARCoPAS: um ambiente para a Recuperação Cooperativa do Projeto Arquietetônico de Sistemas" - M. C. Cavalcanti & M. Borges, VIII Simpósio de Engenharia de Software, Curitiba, outubro/1994;

[ChCr90] "Reverse Engineering and Design Recovery: A Taxonomy" - E. Chikofsky & J. Cross II, IEEE Software, janeiro/1990;

[Chik92] "Untying the Spaghetti: An expert picks the tools" - E. J. Chikofsky, Datamation ("Tips on Reengineering Redundant Software"), 15 de abril de 1992;

[Chik90] "CASE & Reengineering: From Archeology to Software Perestroika" - E. Chikofsky, IEEE Proceedings da 12a Conf. Internacional de Engenharia de Software, 1990;

[ChSc90] "Extracting and Restructuring the Design of Large Systems" - Song Choi & Walt Scacchi, IEEE Software, janeiro/1990;

[CoBe87] "gIBIS: A Hypertext Tool for Team Design Deliberation" - J. Conklin & M. Begeman, Hypertext'87, novembro/1987;

[CoYo92] "Análise Baseada em Objetos" - P. Coad & E. Yourdon, Campus, 1992;

[CoYo93] "Projeto Baseado em Objetos" - P. Coad & E. Yourdon, Campus, 1993;

[DeJe91] "Lotus Notes at Work" - D. DeJean & S. B. DeJean, Lotus Books Business Solution Series, 1991;

[DeGa87] "A Foundation for the Study of Group Decision Support Systems" - G. DeSanctis & B. Gallupe, Management Science, Vol. 33, num. 5, maio/1987;

[DeMa79] "Structural Analysis and System Specification" - T. DeMarco, Yourdon Inc., 1979;

[DeRi93] "Toward Computer-Supported Concurrent Software Engineering" - P. Dewan & J. Riedl, IEEE Computer, janeiro/1993;

[EdMu93] "RECAST: Reverse Engineering from COBOL to SSADM Specification" - H. Edwards & M. Munro, Proceedings of Working Conference on Reverse Engineering, Baltimore, MD, 1993;

[ElGiRe91] "Groupware: some issues and experiences" - C. Ellis & S. Gibbs & G. Rein, Communications of ACM, vol. 34, n. 1, janeiro/1991;

[Fair85] "Software Engineering Concepts" - R. Fairley, McGraw Hill Book Company, 1985;

[FlMu88] "Redocumenting Software Systems using Hypertext Technology" - N. Fletton & M. Munro, Proceedings of the Conference on Software Maintenance, 1988;

[Foli93] "Folio VIEWS™ Personal Electronic Publishing Software" - Folio Corporation, 1993;

[GaLy91] "Using Program Slicing in Software Maintenance" - D. B. Gallagher & J. R. Lyle, IEEE Trans. on Software Engineering, agosto de 1991;

[GaSa79] "Structured System Analysis: Tools and Techniques" - C. Gane & T. Sarson, Prentice Hall Inc., 1979;

[GaSc87] "On Designing Intelligent Hypertext Systems for Information Management in Software Engineering" - P. Garg & W. Scacchi, Hypertext'87, novembro/1987;

[GaSc90] "A Hypertext System to Manage Software Life-Cycle Documents" - P. Garg & W. Scacchi, IEEE Software, maio/1990;

[Ghez91] "Fundamentals of Software Engineering" - Carlo Ghezzi & Mehdi Jazayeri & Dino Mandrioli, Prentice Hall International editions, 1991;

[Gibb89] "CSCW and software engineering" - S. Gibbs, Workshop on Object-Oriented Development, Université de Genève, julho/1989;

[Gorl91] "Techniques for application software maintenance" - N. Gorla, Information and Software Technology, vol 33, Num 1, janeiro/fevereiro 1991;

[Grud91a] "CSCW: The Convergence of Two Development Contexts" - J. Grudin, CHI'91 Human Factors in Computer Systems, ACM, 1991;

[Grud91b] "Obstacles to user involvement in software product development, with implications for CSCW" - J. Grudin, Int. Journal of Man-Machine Studies, vol. 34, 1991;

[Grud94] "Computer-Supported Cooperative Work: History and Focus" - J. Grudin, Computer, maio/1994;

[Joha88] "Groupware: Computer Support for Business Teams"- R. Johansen, The Free Press: Series in Communication Technology an Society, 1988;

[Koza90] "The 'Catch 22' of Re-Engineering" - W. Kozaczynski, IEEE Proceedings da 12a Conf. Internacional de Engenharia de Software, 1990;

[KuRi70] "Issues as Elements of Information Systems" - W. Kunz & H. Rittel, Working paper #131, Institute of Urban and Regional Development, University of California at Berkeley, 1970;

[Lafu90] "Panel on Software Re-Engineering" - G. Lafue, IEEE Proceedings da 12a Conf. Internacional de Engenharia de Software, 1990;

[Land93] "Hypertext and Collaborative Work: The Example of Intermedia (excerpts)" - G. Landow, Readings in Groupware and CSCW, ed. R. Baecker, Morgan-Kaufmann, 1993;

[Lee90] "SIBYL: A Tool for Managing Group Decision Rationale" - J. Lee, Proceedings of CSCW'90, outubro/1990;

[LeFr91] "Re-Engenharia de Software, um Estudo de Caso" - J. C. Leite & A. P. Franco, V Simpósio Brasileiro de Engenharia de Software, julho/1991;

[LePr90] "Design Recovery A Multi-Paradigm Approach" - J. Leite & A. Prado, 1st International Workshop on Software Reusability, Dormund, Alemanha, julho/1991;

[Meyr86] "Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and applications Framework", N. Meyrowitz, OOPSLA'86 Proceedings, 1986;

[MuTi92] "A Reverse Engineering Environment Based on Spatial and Visual Software Interconnection Models" - H. Müller & S. Tilley & M. Orgun & B. Corrie & N. Madhavji, SIGSOFT'92: Proceedings of the 5th ACM SIGSOFT Symposium on Software Development Environments, VA, dezembro/1992;

[Neuw90] "Issues in the Design of Computer Support for Co-authoring and Commenting" - C. Neuwirth & D. Kaufer & R. Chandhok & J. Morris, CSCW'90 Proceedings, 1990;

[Niel93] "Hypertext and Hypermedia" - J. Nielsen, Academic Press, 1993;

[Nuna91] "Eletronic Meeting Systems to Support Group Work" - J. Nunamaker & A. Dennis & J. Valacich & D. Vogel & J. George, Communications of the ACM, vol. 34, n. 7, julho/1991;

[OHTr94] "RE-Analyzer: From source code to structured analysis" - A. O'Hare & E. Troan, IBM Systems Journal, vol 33, num 1, 1994;

[Olso92] "How a Group-Editor Changes the Character of a Design Meeting as well as its Outcome" - J. Olson, G. Olson, M. Storrøsten, M. Carter, CSCW'92 Proceedings, novembro/1992;

[Olso93] "CSCW - Research Issues for the 90's" - J. Olson & S. Card & T. Landauer & G. Olson & T. Malone & J. Legget, User Interface Strategies'94, University of Maryland, dezembro/1993;

[Oman90] "CASE Analysis and Design Tools" e "Maintenance Tools" - P. Oman, IEEE Software, Maio/1990;

[PoBr88] "Recording the Reasons for Design Decisions" - C. Potts & G. Bruns, X International Conf. on Software Engineering, 1988;

[PrLuLe91] "Registro de Decisões e Justificativas de Desenho em Software Projetado com a Metodologia JSD" - A. Prado & C. Lucena & J. C. Leite, Monografias em Ciência da Computação No. 10/91, Pontifícia Universidade Católica do Rio de Janeiro, 1991;

[Pres87] "Software Engineering, A Practioner's Approach" - Pressman, R. S.; McGraw Hill Book Company, 1987;

[ReEl91] "rIBIS: a real-time group hypertext system" - G. Rein & C. Ellis, International Journal of Man-Machine Studies, vol. 34, 1991;

[Roch87] "Análise e Projeto Estruturado de Sistemas" - A. Rocha, Ed. Campus, 1987;

[Roch92] "Experiências no Desenvolvimento de Software Educacional" - A. Rocha & C. Asanome & F. Campos & G. Campos & G. Travassos & J. Souza & K. Sequerra & M. Stahl & N. Santos & R. Costa, III Simpósio Brasileiro de Informática na Educação, SBC, setembro/1992;

[RuCl93] "The Representation Problem in Reverse Engineering" - S. Rugaber & R. Clayton, proceedings da Working Conference on Reverse Engineering, Baltimore, MD, maio/1993;

[SaMyGü92] "Hypertext for Software Engineering: Automatic Conversion of Source Code and Its Documentation into an Integrated Hypertext" - F. Sarre & A. Myka & U. Güntzer, Proceedings of the Intern. Conference DEXA'92, Valência, Espanha, Springer-Verlag Wien, N.Y., 1992;

[ScPl89] "Cross References are Features" - R. Schwanke & M. Platoff, Proceedings of 2nd International Workshop on Software Configuration Management, outubro/1989;

[SeBa91] "Analyzing Error-Prone System Structure" - Richard Selby & Victor Basili, IEEE Transactions on Software Engineering, vol 17, num 2, fevereiro/1991;

[SeWaCh93] "Challenges to the Field of Reverse Engineering" - P. Selfridge & R. Waters & E. Chikofsky, proceedings da Working Conference on Reverse Engineering, Baltimore, MD, maio/1993;

[ShHa94] "Argumentation-Based design rationale: what use at what cost?" - S. Shum & N. Hammond, International Journal of Human-Computer Studies, n.40, 1994;

[Simo87] "Introducing Software Engineering" - G. Simons, NCC Publications,1987;

[Somm85] "Software Engineering" - I. Sommerville, 2nd edition, Workingham, England: Addison-Wesley, 1985;

[Stef87] "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings" - M. Stefik & G. Foster & D. Bobrow & K. Kahn & S. Lanning & L. Suchman, CACM, vol. 30, n. 1, janeiro/1987;

[Walk85] "Document Examiner: Delivery Interface for Hypertext Documents" - J. Walker, Hypertext'87, novembro/1987;

[Whit94] "Using the Booch Method - A Rational Approach" - I. White, Benjamin Cummings, 1994;

[YaCo90] "Report on a Development Project Use of an Issue-Based Information System" - B.Yakemovic & J. Conklin, Proceedings da CSCW'90, outubro/1990;

[YaMeDa85] "Reading and Writing the Electronic Book" - N. Yankelovich & N. Meyrowitz & A. van Dam, Multimedia Communications, outubro/1985;

[YoCo79] "Structured Design" - E. Yourdon & L. Constantine, Prentice Hall, Englewood Cliffs, N.J., 1979;

[Your79] "Managing the Structured Techniques" - E. Yourdon, Prentice Hall, Englewood Cliffs, N.J., 1979;

[Your89a] "RE-3 Re-engineering, Restructuring, Reverse Engineering E. Yourdon, Revista American Programmer Vol.2 No.4 abril/89 e No.6 junho/89;

[Your89b] "Revisões Estruturadas" - E. Yourdon, editora Campus, 1989.