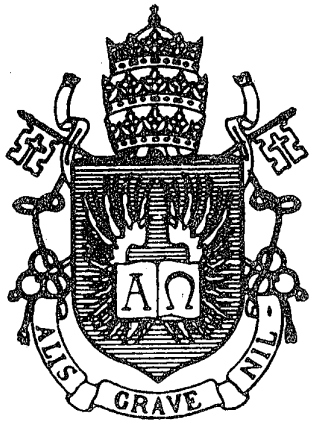

PUC

Série: Monografias em Ciência da Computação

Nº 8/85

MOSAICO - ESTRUTURADO
REQUISITOS

por

Arndt von Staa

Departamento de Informática

)NTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP-22453

RIO DE JANEIRO - BRASIL

PUC/RJ - DEPARTAMENTO DE INFORMÁTICA

Série: Monografias em Ciência da Computação, Nº 8/85

Editor: Paulo A. S. Veloso Novembro, 1985

MOSAICO - ESTRUTURADO

REQUISITOS*

por

Arndt von Staa

*Trabalho parcialmente financiado pela FINEP, pelo CNPq e FIPEC.

Resumo

São definidos no presente documento: os objetivos, a arquitetura e os requisitos do ambiente de apoio ao projeto de sistemas de programação MOSAICO-estruturado. Este sistema está sendo projetado e desenvolvido na PUC/RJ. MOSAICO-estruturado é o primeiro passo visando o desenvolvimento do ambiente de desenvolvimento MOSAICO-geral descrito em [Staa85b].

MOSAICO-estruturado tornará disponíveis ferramentas de definição, de edição, de formatação, de controle de qualidade e de transformação das representações geradas durante o projeto de sistemas de programação. As representações centrais de MOSAICO-estruturado são os diagramas de estrutura organizacional de programas (diagramas estruturados) e de dados (estruturas lógicas).

MOSAICO-estruturado auxiliará na geração de código através de instrumentos de linearização dos diagramas de estrutura organizacional. Os arquivos sequenciais gerados durante a linearização conterão pseudo-código em graus variados de conversão para código fonte de alguma linguagem de programação de alto nível. Em casos extremos (manutenção) estes arquivos poderão conter código fonte diretamente compilável. Com o uso de editores de arquivos (processadores de palavra), o programador poderá converter o pseudo-código em código fonte de alguma linguagem de programação. Cabe observar que MOSAICO-estruturado não está associado a uma linguagem de programação específica.

MOSAICO-estruturado gravita em torno de uma base de software. A base de software é um banco de dados capaz de armazenar todas as informações técnicas (fatos) que ocorrem em um projeto de programa criado e mantido com o apoio de MOSAICO-estruturado. São exemplos de tais fatos: diagramas de estrutura organizacional de programas e de dados; descrições de parâmetros trocados entre módulos; assertivas de entrada e saída dos módulos; pseudo-código ou mesmo código de cada um dos módulos; texto descritivo de cada um dos módulos.

É pressuposto que o leitor do presente documento tenha lido o documento Staa, A.v. Projeto MOSAICO: Definição do Projeto; Informática, PUC/RJ; 1985 [Staa85b].

1. Introdução

O processo de desenvolvimento de sistemas de programação tem deixado muito a desejar, tanto no que tange á produtividade das equipes, quanto no que tange á qualidade dos produtos desenvolvidos. A causa primária para estas deficiências è o processo quase artesanal do desenvolvimento de software. Através do emprego de ferramentas automatizadas è esperado que se consiga, a um sò tempo, aumentar tanto a produtividade das equipes como a qualidade dos sistemas de programação criados com o auxìlio destas ferramentas.

Ferramentas integradas de apoio ao desenvolvimento consituem um ambiente de desenvolvimento de sistemas de programação. Este è essencialmente um sistema CAD/CAM ("Computer Aided Design / Computer Aided Manufacturing") apoiando a especificação, o projeto, a implementação, o controle de qualidade e a manutenção de sistemas de programação.

Para serem eficazes, ferramentas devem formar um todo harmonioso e consistente. Para tal è necessàrio que estas ferramentas sejam projetadas de modo que o conjunto participe de todo o ciclo de vida do sistema de programação. Em adição, o conjunto de ferramentas de apoio ao desenvolvimento de software deve auxiliar o desenvolvimento de sistemas quaisquer, não se limitando a situações de laboratòrio ou a aplicações comerciais, tal como ocorre com a quase totalidade de ferramentas disponìveis hoje.

MOSAICO [Staa85b] è um ambiente de desenvolvimento de sistemas de programação. Ele parte do princìpio que o desenvolvimento de software è realizado através de transformações sucessivas, produzindo e/ou alterando representações (documentos). Uma representação è um veìculo de armazenamento e de comunicação de informações. Sàn exemplos de representações: os documentos externos destinados aos usuàrios do sistema de programação, os documentos internos destinados ao pessoal tècnico, a base de software que armazena as informações de projeto, os textos e diagramas apresentados em telas de computador, etc.

O desenvolvimento de MOSAICO serà realizado em etapas. A primeira destas etapas è desenvolver MOSAICO-estruturado que vem a ser o objeto do presente documento.

MOSAICO-estruturado é um ambiente de desenvolvimento (conjunto integrado de ferramentas automatizadas) visando apoiar a criação, manutenção e o controle de qualidade de projetos de sistemas de programação. MOSAICO-estruturado será capaz de lidar com instrumentos modernos de projeto de programas, tais como estruturas funcionais (abstrações de dados), programas distribuídos, corotinas, tratamento de exceções, modularização de programas, estruturação de código, pseudo-código.

Durante o projeto lógico e o projeto físico de programas são gerados diagramas de estrutura organizacional de programas e de dados (ver figura 2.1). Diagramas de estrutura organizacional formam a ponte entre a anàlise e a programação. Estes diagramas são utilizados, também, durante a manutenção, sempre que as alterações afetem a organização do programa ou dos dados.

Diagramas de estrutura organizacional destinam-se a analistas, projetistas e programadores. Estes diagramas são as linguagens de representação fundamentais para a criação sistemàtica de programas. As principais metodologias de projeto de programas em uso hoje em dia utilizam diagramas de estrutura organizacional [DeMarco78, Gane83, Jackson75, Page-Jones80, Yourdon79]. Estes diagramas podem, também, ser utilizados em outros contextos, tais como: planejamento e acompanhamento do desenvolvimento de sistemas de programação; especificação da organização de documentos; especificação de estruturas de dados; especificação de restrições dinâmicas de uso de funções; etc.

O ambiente de desenvolvimento MOSAICO-estruturado manterà uma base de software capaz de armazenar todos fatos que ocorrem em projetos de programas. São exemplos de tais fatos: os componentes das estruturas organizacionais de programas e de dados; o empacotamento das organizações modulares; o pseudo-código ou código associado a cada uma dos mòdulos; os parâmetros passados e recebidos por cada um dos mòdulos; as assertivas de entrada e saída dos mòdulos; etc.

MOSAICO-estruturado è composto essencialmente por dois editores. O primeiro permite criar, alterar, empacotar e explorar estruturas organizacionais. O tipo do conteúdo destas estruturas è, em princìpio, qualquer, por exemplo interconexão de mòdulos, lògica interna de mòdulos, estruturas de dados, estruturas de tela, estruturas de documentos, estruturas de tarefas etc.

O segundo editor permite criar, alterar e explorar texto. Este editor è essencialmente um processador de palavras adaptado á base de software. O editor de texto não prevê estrutras internas ao texto. Caso existam tais estruturas, elas deverão ser definidas externamente ao editor de texto. Por exemplo, o dicionário de dados poderia ser definido em termos de uma estrutura de dados onde os campos são textos.

Além de editores, MOSAICO-estruturado disporá de transformadores de informação capazes de converter estruturas organizacionais em texto sequencial (linearizadores). Tipicamente a linearização de uma estrutura organizacional produz uma sequência de pseudo-código. Após ter obtido este pseudo-código, o usuário poderá transformá-lo em código com o auxílio de um editor de arquivos. Futuras versões de MOSAICO prevêem a incorporação de editores de arquivo, permitindo, assim, aos linearizadores a geração automática de código.

Através do uso de MOSAICO-estruturado espera-se, a um só tempo, obter ganhos significativos tanto em produtividade como em qualidade resultante. Tais ganhos trazem diversos benefícios para os usuários de MOSAICO-estruturado, tais como: aumentar a capacidade de produção dos engenheiros de software; menores custos de desenvolvimento; maior confiabilidade dos sistemas e, consequentemente, menores danos devidos ao mau funcionamento do sistema; menos perdas de oportunidade; redução da demanda reprimida, etc.

2. O processo de projetar sistemas de programação

Nesta seção será focalizada a etapa de projeto de sistemas de programação. Esta é a etapa a ser apoiada por MOSAICO-estruturado. Ao leitor interessado em mais detalhes recomendamos as referências [Freeman83b, Jackson75, Page-Jones80].

Uma etapa crucial para o desenvolvimeto de sistemas de programação corretos e confiàveis è o projeto do sistema. MOSAICO-estruturado visa especificamente apoiar esta etapa do processo de desenvolvimento de software. A etapa de projeto de sistemas de programação subdivide-se nas seguintes sub-etapas:

arquitetura: o objetivo desta etapa è transformar a especificação de requisitos e a especificação funcional em uma organização de solução. Para tal è necessàrio, entre outros: Identificar os diferentes programas a serem desenvolvidos. Determinar o suporte operacional necessàrio para o conjunto destes programas, e.g. biblioteca de programas, sistema de comunicação de dados, sistema de gerência de banco de dados, etc. Determinar o ambiente operacional para cada um dos programas, e.g. computador especifico, sistema operacional, rede de teleprocessamento etc. Determinar a interface dos programas entre si, e.g. protocolos, arquivos temporàrios, base de dados etc. Determinar a interface entre os programas e o seu ambiente aplicacional, e.g. on-line, batch etc. Determinar a interface entre usuàrios e programas. Determinar a estratègia de teste a ser utilizada para efetuar o controle de qualidade operacional do sistema.

projeto lógico: o objetivo desta etapa è transformar a arquitetura e a especificação funcional em projetos estruturados de programas capazes de resolver o problema em algum ambiente de programação. Para tal são necessàrios, entre outros: Determinar os algoritmos a serem utilizados pelo programa. Determinar a organização modular do programa. Determinar as estruturas de dados a serem utilizadas pelo programa. Determinar o modo de interação do programa com seu ambiente de suporte. Determinar o modo de interação do programa com o usuàrio, e.g. projeto de telas, formulàrios etc. Determinar os casos teste a serem utilizados para o controle de qualidade operacional do programa.

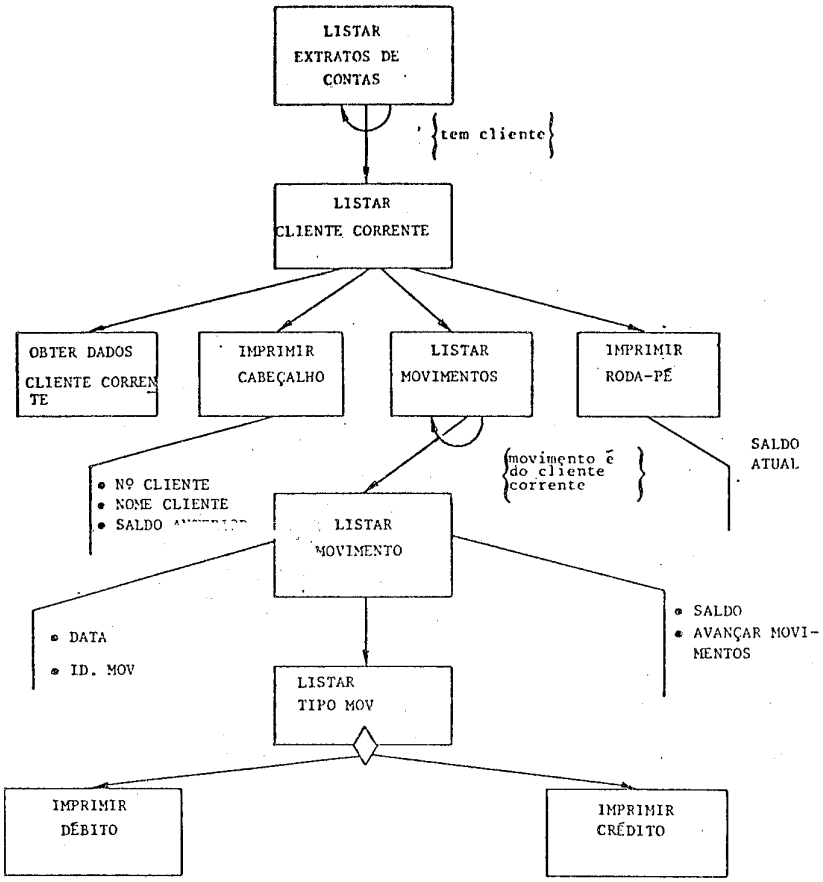

Figura 2.1 Ilustração de diagrama de estrutura organizacional de programa.

projeto físico: o objetivo desta etapa é transformar o projeto lógico em um projeto corretamente programável e viável no ambiente de execução onde residirá o programa. Para tal são necessários, entre outros: Transformar o projeto lógico em um projeto viável no ambiente operacional específico onde residirá o programa. Determinar os módulos físicos que comporão o programa. Determinar a sequência de integração destes módulos físicos. Determinar os formatos da interação do usuário com o programa, e.g. formatação de tela, teclas de controle, texto de mensagens etc. Determinar os casos teste a serem utilizados para comprovar a qualidade operacional de cada um dos módulos físicos.

A principal linguagem de representação utilizada durante o projeto de um programa è o diagrama de estrutura organizacional, ilustrada na figura 2.1. Esta linguagem descreve os relacionamentos hieràrquicos existentes entre os componentes de um programa, e.g. módulos, blocos de controle, etc. Esta mesma linguagem è utilizada para descrever os relacionamentos hieràrquicos existentes entre os componentes de uma estrutura de dados.

3. Objetivos de MOSAICO-estruturado

O objetivo de MOSAICO-estruturado è apoiar a criação e a manutenção de projetos lògicos e fisicos de programas.

Projetos de programas são constituidos por estruturas organizacionais de programas e de dados, definição do empacotamento das estruturas organizacionais formando componentes fisicos, descrição de cada um dos componentes destas estruturas, descrição das interfaces entre componentes, definição dos requisitos gerais das estruturas e de cada um dos componentes, descrição da organização de implementação de cada um dos componentes (pseudo-còdigo), etc. MOSAICO-estruturado utilizarà linguagens de representação gràficas (diagramas de estrutura organizacional) como instrumentos bàsicos de interação com o usuàrio. Note que o usuàrio de MOSAICO serà tipicamente um engenheiro de software.

Para atingir este objetivo, o sistema MOSAICO-estruturado deverà satisfazer os seguintes requisitos primàrios:

i- criar e manter uma base de software distribuida contendo todos os fatos correspondentes aos diversos projetos dos programas de uma dada instalação;

ii- editar a base de software a partir de qualquer representação suportada, em particular a partir de diagramas de estrutura organizacional. A edição serà efetuada utilizando teclado e tela de micro-computador;

iii- detalhar qualquer componente de uma dada estrutura organizacional atravès de:

a- explosão em outro diagrama de estrutura organizacional;

b- identificação do tipo de controle de cada um dos componentes da estrutura organizacional (por exemplo: repetição, seleção, obrigatòrio, etc.);

c- identificação da interface do componente com os demais componentes (ex. parâmetros de entrada e saida);

d- identificação das assertivas de entrada e saída e das restrições de integridade;

e- descrição dos requisitos de cada um dos componentes através de textos (por exemplo: hipóteses, objetivos, textos descritivos, etc.)

f- descrição da implementação do componente através de textos (por exemplo: pseudo-código, pseudo-declarativo, código, etc.);

iv- apoiar o empacotamento da estrutura organizacional (empacotamento: é a definição de uma agregação lógica ou física de um sub-conjunto dos componentes de uma estrutura organizacional);

v- apoiar a linearização da estrutura organizacional, produzindo arquivos sequenciais contendo:

a- caso estrutura organizacional de dados, pseudo declarativos;

b- caso estrutura organizacional de programas, pseudo código;

vi- produzir, em papel, a representação de projetos, utilizando uma impressora gráfica serial ("raster plotter");

vii- permitir ao engenheiro de software desenvolver e manter projetos diretamente em MOSAICO, em princípio sem requerer a prévia confecção de diagramas detalhados (prancheta eletrônica).

4. Arquitetura de MOSAICO-estruturado

Os diagramas a seguir esboçam a arquitetura funcional do sistema MOSAICO-estruturado.

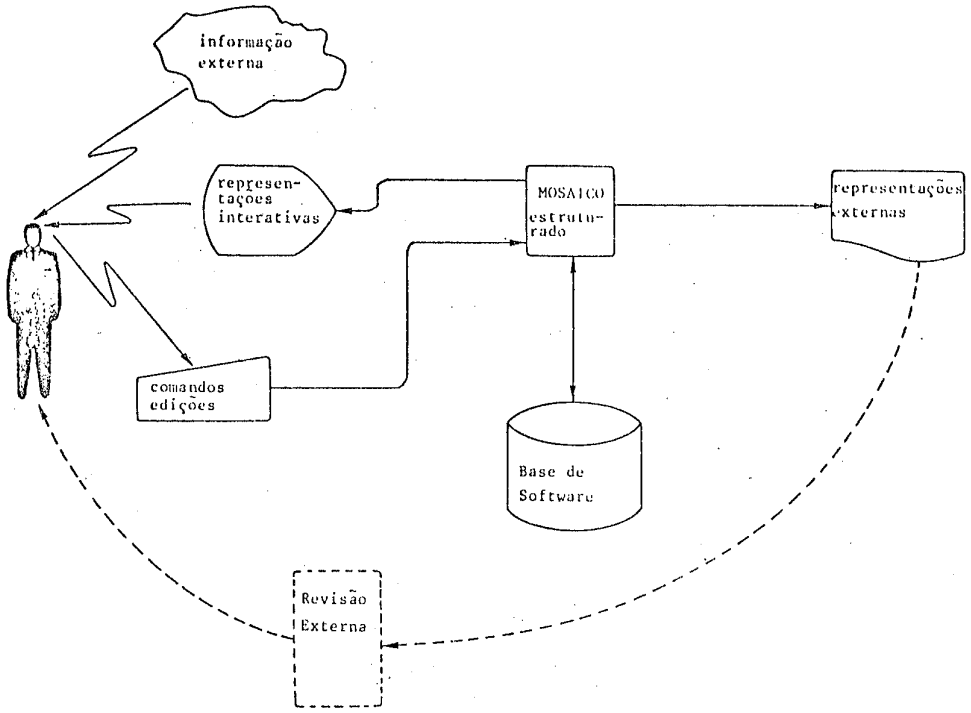

Figura 4.1 Diagrama de contexto de MOSAICO-estruturado

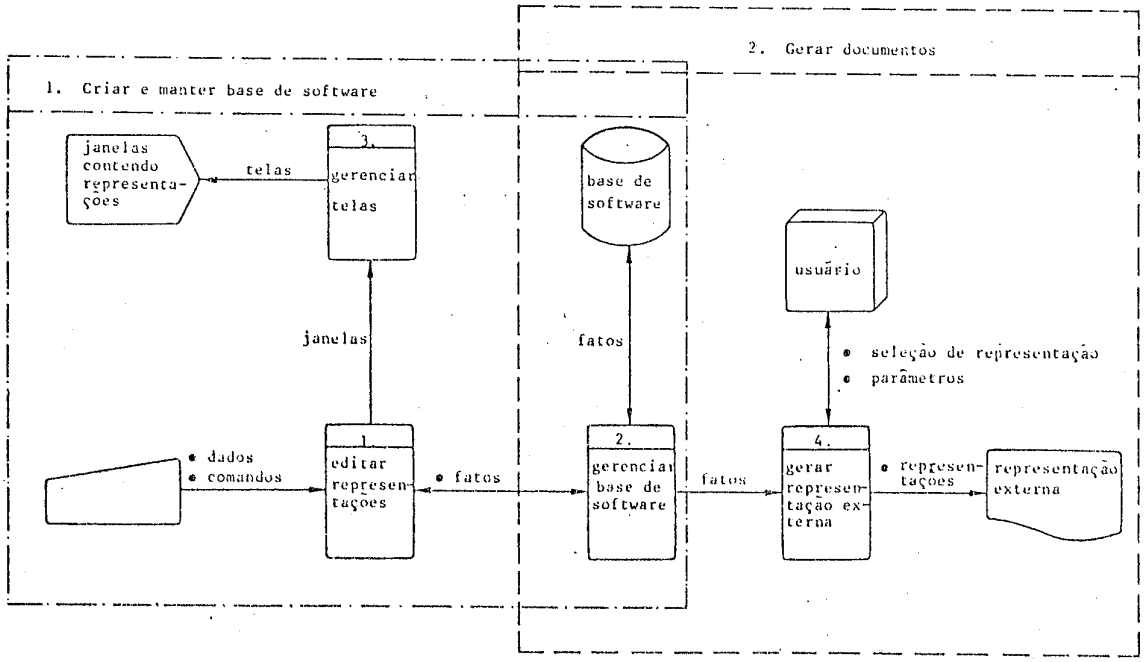

Figura 4.2 Fluxo geral de MOSAICO-estruturado

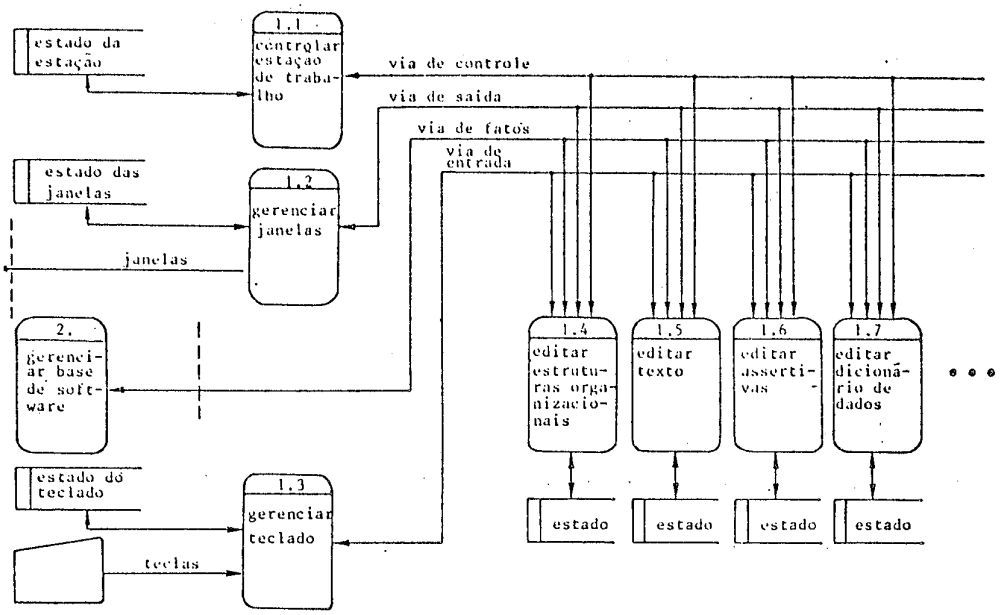

Figura 4.3 Decomposição de 1. Editar representações.

Descrição sumária dos módulos:

1.1 Controlar estação de trabalho

Este módulo controla o estado geral da estação de trabalho e efetua a transição entre os diferentes editores.

1.2 Gerenciar janelas

Este módulo contêm as operações de saída dirigidas ao vídeo do micro-computador.

1.3 Gerenciar teclado

Este módulo contêm as operações de leitura a partir do teclado do micro-computador.

1.4 Editar estruturas organizacionais

Este módulo cria, altera e explora as diversas estruturas organizacionais que perfazem um determinado projeto de programa.

1.5 Editar texto

Este módulo é um processador de palavras vinculado ao MOSAICO.

1.6 Editar assertivas

Este módulo cria, mantêm e explora as assértivas de entrada e saída de cada um dos componentes de um projeto de programa.

1.7 Editar dicionário de dados

Este módulo cria, mantêm e explora as definições dos dados manipulados pelos diversos componentes de um projeto de programa.

2. Gerenciar base de software

Este módulo cria, mantêm e acessa os diversos dados e relacionamentos armazenados na base de software.

5. Requisitos do sistema de arquivos da base de software.

Nesta seção serão definidos os requisitos do sistema de arquivos que implementa a base de software.

A base de software armazenará todos os dados correspondentes aos diversos projetos de uma instalação. Cada um destes projetos possuirá um tamanho próprio. Do ponto de vista de MOSAICO-estruturado, este tamanho poderá ser qualquer. Em um extremo, um projeto corresponderá a um módulo de programa. Em outro extremo, um projeto corresponderá a um sistema completo. Através da hierarquização, tornar-se-á possível desenvolver um sistema grande através de diversos projetos pequenos (tarefas), integrando-os através de um projeto integrador.

Independente da organização lógica da base de software, esta deverá satisfazer aos seguintes requisitos:

1- A base de software será particionada em cadernos. Cada caderno conterá todos os fatos e itens que constituem uma unidade de projeto. Um item é um elemento atômico de infomação sobre o projeto. Um fato é um elemento composto de informação sobre o projeto. Um relacionamento é uma vinculação entre um fato e outros fatos e/ou itens, podendo ou não possuir informação associada. Cada fato é constituído por um conjunto de zero ou mais itens, um conjunto de zero ou mais fatos e um conjunto de relacionamentos entre estes itens e fatos. Estes tres conjuntos formam uma decomposição coesa com o fato.

2- Cada caderno corresponderá a um item de configuração. São exemplos de itens de configuração: programas de um sistema, módulos compilados independentemente, módulos de biblioteca, pacotes de biblioteca, projetos genéricos capazes de serem instanciados para resolverem determinado problema específico etc.

3- Itens de configuração (cadernos) poderão ser combinados com outros itens de configuração. Através da composição de itens de configuração conseguir-se-á desenvolver sistemas constituidos por diversos programas e/ou módulos de programa.

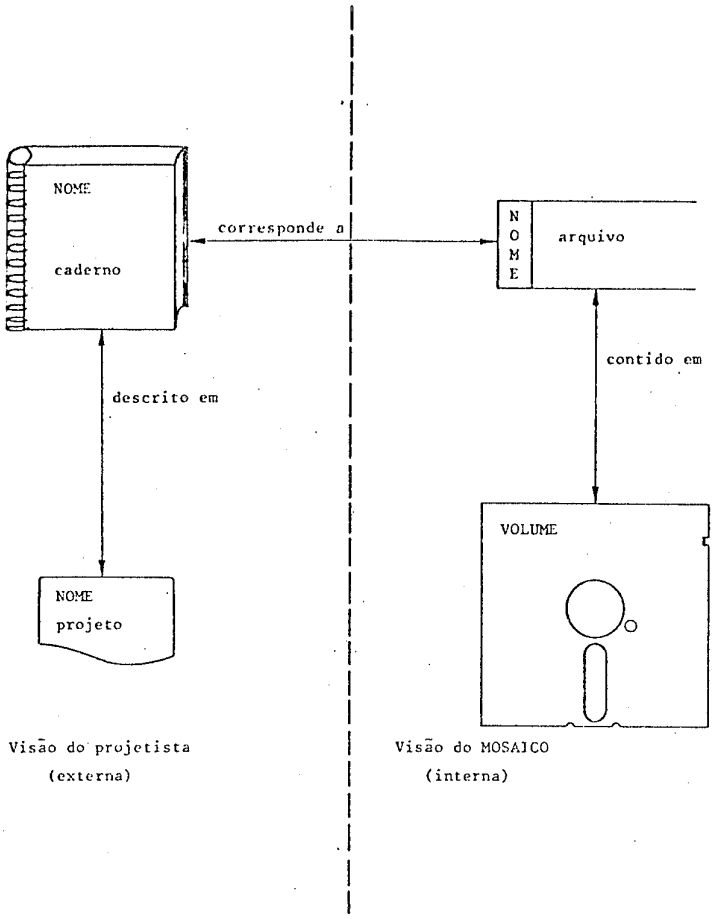

Figura 5.1 Arquitetura básica do sistema de arquivos.

4- Cada caderno estará dedicado a um projetista ou programador. Facilita-se assim a administração dos diferentes projetos, bem como o intercâmbio de informações entre diferentes projetistas trabalhando em comum em um sistema constituido por vários sub-projetos (tarefas).

5- Cada caderno estará contido em um único arquivo. Cada arquivo estará gravado em um único volume (ex. disquete). Um mesmo volume poderá conter diversos cadernos.

6- A base de software manterá um índice de cadernos. Através do índice de cadernos poder-se-á localizar qualquer caderno. O índice deverá ser capaz de identificar tanto o caderno principal, como o caderno backup. Para cada ca-

derno o índice de cadernos fornecerá o nome do caderno, o nome do arquivo físico onde se encontra este caderno, o nome do volume onde se encontra este arquivo, o estado de configuração do caderno, as datas de criação e de modificação, além de outras informações a serem identificadas no futuro.

7- A base de software deverá ser capaz de suportar relacionamentos que vinculem fatos e/ou itens constantes em diferentes cadernos. Desta forma será possível criar e manter projetos agregadores de sub-projetos.

8- A base de software deverá ser capaz de suportar operações de cópia dos fatos contidos em um caderno para outro caderno.

9- A base de software deverá prover utilitários para a confecção de cópias backup de cadernos, bem como para a restauração de cadernos a partir destas cópias backup.

10- A base de software deverá prover utilitários para a verificação da correção estrutural das informações contidas em cada caderno.

11- A qualquer instante poder-se-á proteger a base de software contra alterações indevidas. Estas proteções serão seletivas e poderão ser revogadas.

6. Requisitos do acesso lógico à base de software.

Nesta seção será definido o método de acesso aos fatos e itens contidos em um caderno da base de software.

Independente do modo de armazenamento dos fatos e itens de um caderno, o acesso a esses fatos e itens deverá satisfazer aos seguintes requisitos:

1- Cada caderno é formado por um conjunto de folhas eletrônicas. Cada folha possui dimensões, em princípio, infinitas.

2- Cada folha eletrônica conterá os itens, fatos e relacionamentos requeridos por uma única linguagem de representação. Ou seja, cada folha possui um determinado tipo de folha. Este tipo estabelecerá a vinculação dos itens, fatos e relacionamentos da base de software com esta linguagem de representação.

3- Um mesmo item, fato ou relacionamento poderá figurar em diferentes folhas eletrônicas. Estas folhas poderão estar em um mesmo caderno ou em diferentes cadernos. Ao criar,

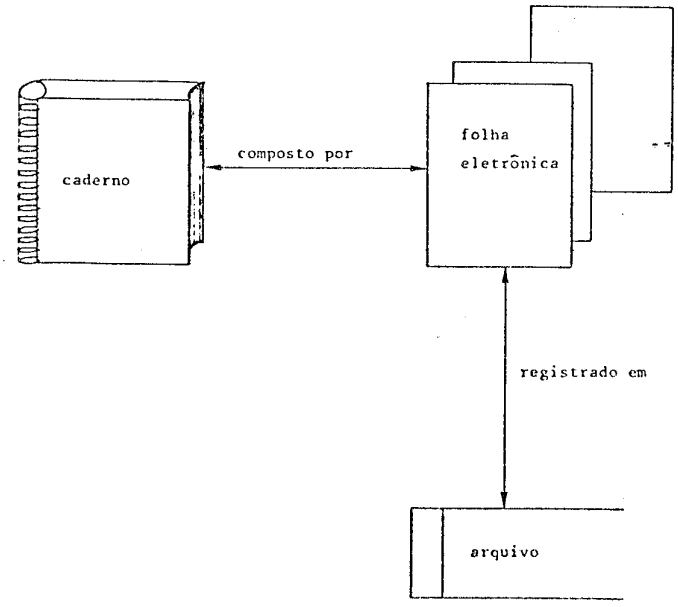

Figura 6.1 Composição de um caderno.

alterar ou delir um elemento de uma folha eletrônica, este elemento será criado, alterado ou delido em todas as folhas eletrônicas em que apareça.

4- Os itens, fatos e relacionamentos existentes em diferentes folhas eletrônicas são conjuntos não comparáveis. Ou seja, caso duas folhas eletrônicas possuam interseção não vazia, não é necessário que uma folha seja sub-conjunto da outra.

5- Em um caderno poderão co-existir diversas folhas de um mesmo tipo. Cada uma destas folhas descreve completamente um único componente do projeto. Por exemplo, caso um projeto possua diversos diagramas de estrutura organizacional, como é o caso em uma estrutura funcional

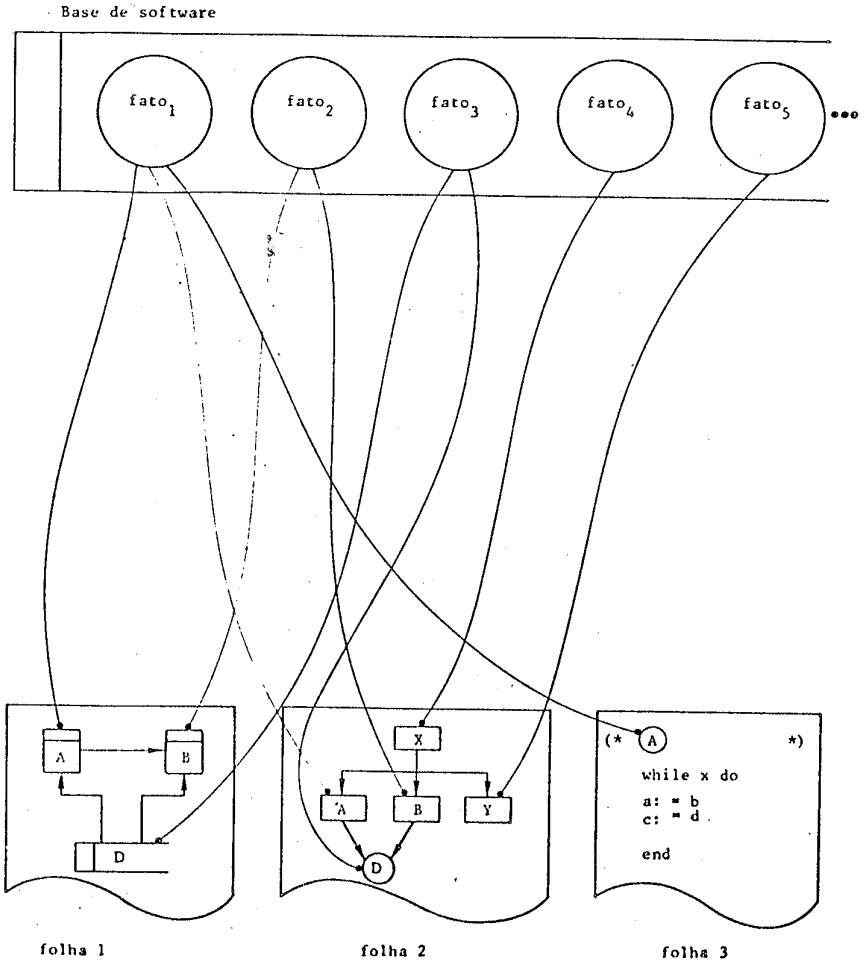

Figura 6.2 Ilustração da relação entre fatos e folhas eletrônicas.

(abstração de dados), cada um destes diagramas de estrutura organizacional estará integralmente contido em uma única folha.

6- Os itens, fatos e relacionamentos contidos em uma folha eletrônica deverão satisfazer uma gramática da folha. Esta gramática estará em correspondência com a linguagem de representação da folha eletrônica. Ou seja, a gramática será selecionada pelo tipo da folha eletrônica.

7- A gramática da folha poderá vincular itens, fatos e relacionamentos que ocorrem em diferentes folhas.

8- Através da gramática da folha será possível verificar a correção e a completeza estrutural dos itens, fatos e relacionamentos que aparecem em uma folha eletrônica.

9- Através da gramática será possível gerar estruturas ainda não interpretadas sempre que necessário em função de uma edição. Desta forma tornar-se-á possível criar e/ou alterar elementos interrelacionados, antes de se dispor de todos os dados deste relacionamento.

7. Requisitos de utilizabilidade

Nesta seção será definido como MOSAICO-estruturado visualizará os fatos e itens contidos na base de software. Uma das características essenciais de MOSAICO-estruturado é a sua facilidade de uso. Para realmente contribuir para um aumento de desempenho da equipe de desenvolvimento, é essencial que a interface homem/sistema seja simples, ágil e poderosa.

Além de ser ameno ao usuário, MOSAICO-estruturado deverá suportar tanto estruturas organizacionais de dados, como suportar estruturas organizacionais de programas.

A lista de requisitos de utilizabilidade não é especificável a priori, uma vez que muitos destes requisitos dependem de "feedback" a ser obtido com o uso de MOSAICO-estruturado. É essencial, portanto, que MOSAICO-estruturado seja de fácil alteração, permitindo assim o ajuste e a experimentação com diferentes modos de interação.

Independente do formato gráfico das linguagens de representação utilizadas, MOSAICO-estruturado deverá satisfazer os seguintes requisitos:

1- MOSAICO exibirá itens, fatos e relacionamentos (elementos) na tela. Esta exibição se dará através de janelas [Goldberg83]. Cada janela estará associada a uma única folha eletrônica. Cada janela exibirá na tela um sub-conjunto dos itens, fatos e relacionamentos contidos nesta folha. O sub-conjunto de elementos a ser exibido dependerá do volume dos elementos contidos na folha eletrônica, das dimensões físicas da janela, do formato de exibição e da posição da janela sobre a folha eletrônica. MOSAICO proverá um amplo elenco de formatadores e de operações de posicionamento da janela sobre a folha eletrônica.

2- Em um determinado instante poderão estar abertas diversas janelas. Os elementos exibidos em diferentes janelas advirão de diferentes folhas eletrônicas. Estas folhas poderão figurar em um mesmo caderno e/ou em diferentes cadernos.

3- MOSAICO implementará a noção de corrência. A cada instante estarão identificados na tela a janela corrente, bem como os itens e fatos correntes. Os elementos correntes serão

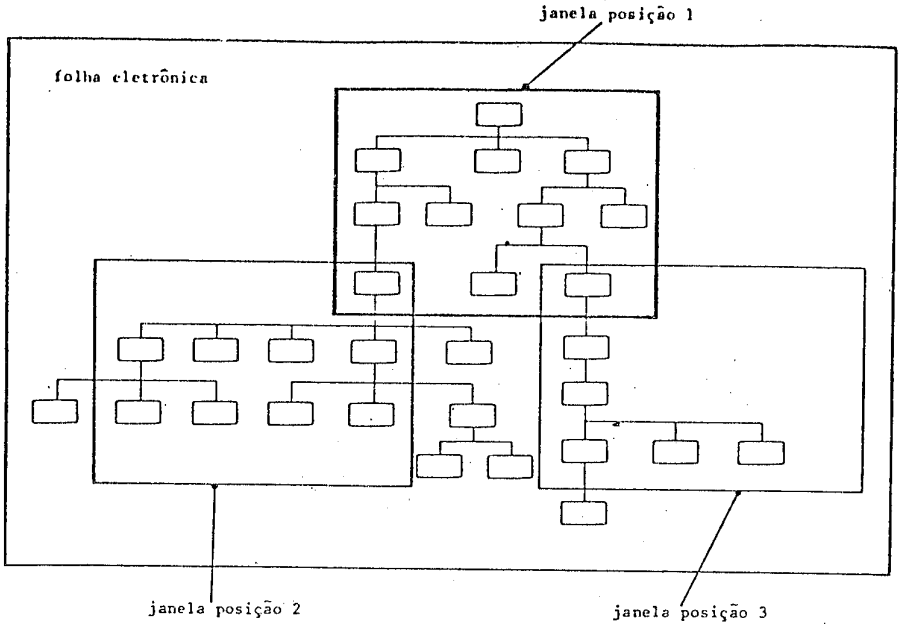

Figura 7.1 Relação entre folhas eletrônicas e janelas.

exibidos de tal forma que se distinguam dos elementos não correntes. Através de um amplo conjunto de operações, o usuário poderá tornar corrente qualquer janela e, nesta, qualquer elemento. Os elementos correntes estarão sempre presentes em alguma janela aberta, no entanto, esta janela poderá não ser a janela corrente.

4- O conjunto de janelas forma um maço de folhas. A gerência de janelas deve dar ao usuário a impressão de estar trabalhando com diversas folhas eletrônicas ao mesmo tempo, sendo que a folha correspondente á janela corrente estará sempre no topo do maço de folhas em uso no momento. Através de operações do usuário será identificada a janela corrente.

5- As folhas eletrônicas correspondentes a janelas abertas, poderão advir de cadernos diferentes, e/ou de um mesmo caderno.

6- Os elementos exibidos na janela corrente estarão em correspondência estrita com os elementos registrados na base de software. Para assegurar a correspondência do conteúdo

da janela com o conteúdo de memória, poderá ser necessário efetuar operações de redesenho do conteúdo da janela corrente.

7- Elementos que figuram em janelas diferentes da janela corrente não estarão necessariamente em correspondência estrita com o conteúdo de memória. Consequentemente, sempre que uma janela for tornada corrente, o seu conteúdo e organização deve ser recalculado.

8- A criação, edição e a exploração do detalhamento de um projeto será feita através de operações de caminhamento, criação, alteração e deleção comandadas pelo usuário. Estarão diponíveis as seguintes operações:

a- navegar na estrutura organizacional;

b- mudar de estrutura organizacional;

c- explorar um módulo através do acesso aos diversos componentes que o constituem;

d- editar todos os elementos do caderno;

e- mover e copiar elementos contidos neste e/ou em outro caderno;

9- O caminhamento poderá ser efetuado tanto através de teclas de movimentação de cursor, como através de posicionamento por consulta a listas de componentes satisfazendo um determinado crítério de escolha. São exemplos de tais listas, a lista dos filhos de determinado módulo, a lista de pais de um módulo, a lista de módulos utilizando um determinado conjunto de parâmetros, etc.

10- Todas as operações e edições serão efetuadas a partir da janela corrente. Esta assegurará a identificação inconfundível das porções que poderão ser afetadas por estas operações ou edições.

11- Serão suportadas operações convencionais de edição, tais como: inserção, eliminação, alteração de elementos; cópia e movimentações de elementos, possivelmente envolvendo diferentes cadernos, etc.

12- Serão suportadas operações menos convencionais, tais como: desfazer o efeito de operações; salvaguarda em caderno "limbo" dos elementos a serem eliminados; empacotamento e desempacotamento de estruturas organizacionais; etc.

13- Edições poderão ser efetuadas em qualquer folha de qualquer caderno, desde que o elemento afetado esteja corrente na janela corrente. O efeito de uma edição será global, ou seja, será tornado aparente em todas as outras representações que utilizem este elemento.

Além de exibir representações no vídeo, MOSAICO será capaz de produzir listagens contendo estas representações. A impressão deve satisfazer os seguintes requisitos:

14- o aspecto gráfico impresso deverá ser semelhante ao aspecto gráfico exibido na tela. No entanto, as dimensões dos gráficos impressos não devem estar sujeitos ás mesmas restrições de dimensão que os gráficos exibidos.

15- as dimensões do papel, horizontal ou vertical, deverá poder ser qualquer, sendo controlado por parâmetro dinâmico.

16- A impressão de estruturas organizacionais poderá ser efetuada nas formas a seguir:

a- página a página. Cada página conterá uma porção da organização modular perfeitamente delimitada. Diagramas grandes utilizarão diversas páginas evidenciando as conexões existentes entre páginas.

b- "papel de parede". Diagramas serão impressos em "tiras" de formulário continuo. As diversas tiras que compõem um diagrama, deverão ser emendadas lado a lado permitindo assim a exibição do diagrama completo.

17- O grau de detalhe do documento a ser impresso será seletivo.

A interação do usuário com MOSAICO se dará através de linguagens de comando. As linguagens de comando utilizadas por MOSAICO devem satisfazer os seguintes requisitos:

18- clareza de identificação do estado corrente. O sistema sempre exibirá toda a informação indicativa do estado corrente e do componente focal no momento. São exemplos de estados: exploração de estrutura, inserção de módulo, exploração do interior de um módulo, etc. São exemplos de componentes focais: o caractere sob o cursor, o módulo em torno do qual se dará a inserção, a porção de texto a ser movida, etc.

19- uniformidade de função. Cada tecla terá associada a si uma única função lógica. Dependendo do estado, a função física a ser efetuada varia. São exemplos de funções lógicas: caminhar para a direita, para a esquerda, para cima, para baixo, inserir, delir, etc. São exemples de funções lógicas: cainhar para módulo á esquerda, caminhar para caractere á esquerda, inserir caractere, inserir módulo, etc.

20- adaptabilidade das teclas. O usuário poderá redefinir a associação entre teclas e funções lógicas.

21- auxílio e mensagens dependem do estado do sistema. As mensagens serão curtas e logicamente vinculadas ao estado do sistema no momento em que a mensagem foi liberada para ser emitida.

22- a qualquer instante o usuário poderá salvar o conteúdo da memória principal, sem que isto provoque uma alteração no estado de corrência do sistema.

8. Restrições

MOSAICO-estruturado impõe uma sèrie de restrições. Estas restrições são:

1- MOSAICO-estruturado será mono-usuário e será implementado em um micro computador compatível com PC-IBM com pelo menos 256K de memória principal, operando com sistema operacional MS-DOS versão 2.10 ou compatível.

2- MOSAICO-estruturado será implementado utilizando a linguagem Modula-2 [Wirth82], compilador Modula-2/86 de Logitech, versão 1.02 ou compatível.

9. Avaliação da utilidade

Nesta seção examinaremos como poderá ser utilizado MOSAICO-estruturado. Examinaremos, ainda, os benefícios trazidos pelo seu uso. Como já foi mencionado, MOSAICO-estruturado é um ambiente que apoia o projeto de programas utilizando técnicas modernas de projeto. MOSAICO-estruturado apoia, também, a transformação de projetos em pseudo-código estruturado compatível com uma linguagem de programação de escolha do usuário.

9.1 Identificação dos usuários de MOSAICO-estruturado

MOSAICO-estruturado destina-se a qualquer analista, projetista ou programador. Apesar de estar sendo desenvolvido em um equipamento compatível com o IBM-PC, os programas desenvolvidos com apoio de MOSAICO-estruturado poderão residir e ser executados em qualquer equipamento, qualquer que seja o seu porte, fabricante e suporte operacional utilizado. Para tal será necessário poder transmitir arquivos sequenciais do micro-computador de desenvolvimento para o computador de residência do programa.

Os programas desenvolvidos com apoio de MOSAICO-estruturado poderão ser quaisquer, tais como aplicações comerciais, aplicações de automação industrial, software básico, aplicações de engenharia, etc. Finalmente, MOSAICO-estruturado não está vinculado a uma linguagem de programação específica, sendo facilmente ajustado a qualquer linguagem de programação procedural (ex. Pascal, COBOL, FORTRAN). Consequentemente, os usuários potenciais de MOSAICO-estruturado abrangem virtualmente todas as equipes que desenvolvem e/ou mantêm sistemas de programação.

MOSAICO-estruturado é o primeiro de uma série de inúmeros passos. A cada passo dispor-se-á de um ambiente mais elaborado e abrangente. Assim é razoável assumir que usuários evoluam junto com MOSAICO passando para versões mais avançadas á medida que estas se tornem disponíveis.

9.2 Criação de um novo programa

A seguir ilustramos, em linhas gerais, como o projetista poderia utilizar MOSAICO-estruturado para criar projetos de programas. Ao criar um novo programa, o projetista:

a- cria um novo caderno

b- define uma estrutura organizacional para o programa

c- define as condições de entrada e saída de cada um dos componentes

d- define parâmetros de entrada e saída de cada um dos componentes

e- adiciona á estrutura organizacional informação de controle de execução: repetições, seleções, etc.

f- empacota a estrutura organizacional formando procedimentos autocontidos ("procedures", "packages" etc.)

g- empacota os parâmetros determinando regiões de COMMON, globais etc. onde ficarão estes parâmetros

h- percorre a estrutura efetuando o controle de qualidade. Esta etapa pode contar com a ajuda de um colega e visa examinar se o comportamento dinâmico projetado corresponde ao comportamento dinâmico esperado

i- lineariza a estrutura produzindo pseudo-còdigo

j- gera còdigo fonte preenchendo lacunas no texto linearizado. São lacunas os pontos onde foi gerado pseudo-còdigo. Utiliza um processador de palavra para este fim

k- gera casos teste utilizando um "propositor estrutural"

9.3 Alterar um programa já existente

A seguir ilustramos, em linhas gerais, como o projetista poderia utilizar MOSAICO-estruturado para manter projetos de programas. Ao alterar um projeto já existente, o projetista:

a- localiza e seleciona o caderno onde se encontra o projeto

b- percorre a estrutura do programa em busca dos pontos a alterar

c- efetua as alterações em cada um dos pontos localizados

d- refaz, se necessário, o empacotamento

e- percorre a estrutura organizacional controlando a qualidade do projeto

f- lineariza, gerando código e/ou pseudo-código. Será possível a geração de código nas regiões não afetadas pela alteração. Nas demais, será gerado pseudo-código

g- completa o programa preenchendo lacunas no texto linearizado. Lacunas existem onde se encontra pseudo-código modificado pela alteração. Utiliza um processador de palavra para este fim

h- gera casos teste utilizando um propositor estrutural para as porções alteradas.

9.4 Benefícios para o usuário

Os principais benefícios auferidos pelo uso de MOSAICO-estruturado são:

1- Aumento da produtividade da equipe técnica

- redução do esforço de comunicação analista/programador. Em virtude de estar-se utilizando uma linguagem de representação formalizada, torna-se possível a passagem do projeto para o programador, sem requerer uma descrição e/ou apresentação pormenorizada e volumosa. Em adição, a linguagem de representação utilizada permite a geração automática de seqüências de pseudo-código próximas á linguagem de programação a ser utilizada e onde este pseudo-código corresponde fielmente ao projeto. As seqüências de pseudo-código determinam a estrutura do programa. Sobra, assim, ao programador somente a tarefa de converter o pseudo-código em código compilável (programação por preenchimento de lacunas).

- redução do esforço de documentação. Em virtude de se estar utilizando formatadores a partir da base de software,

torna-se possível a geração de documentos de boa qualidade de acabamento diretamente a partir do trabalho criativo efetuado pelo projetista. Cabe observar que o conjunto de documentos usualmente contêm redundâncias entre si. Redundâncias ocorrem quando um mesmo fato é registrado em diversos lugares diferentes. Em MOSAICO-estruturado as redundâncias não oneram a geração da documentação, uma vez que o projetista define cada fato uma única vez e os formatadores se encarregam de apresentá-los devidamente formatados e diagramados em todos os diferentes documentos onde porventura ocorram.

- redução do esforço de manutenção. Em virtude do uso de editores vinculados á linguagem de representação externa utilizada é facilitada a procura dos pontos onde o programa deverá ser alterado. É facilitada, ainda, a percepção do impacto de uma alteração. Como já foi mencionado, as alterações efetuadas na base de software são automaticamente refletidas em todos os documentos gerados a partir desta base de software. Finalmente, a sequência de alterações e de controle das alterações poderá ficar circunscrita às porções afetadas pela manutenção. Isto tudo contribui para uma significativa redução de esforço dispendido a cada modificação do programa. Cabe observar que em grande parte das instalações o esforço dispendido em manutenção ultrapassa 50% de todo o esforço consumido com software.

- viabilização da reutilização de projetos anteriores. Em consequência da facilidade de alteração, torna-se possível criar um novo projeto a partir de um outro semelhante já existente, diminuindo o esforço dispendido em desenvolvimento.

- redução do esforço de depuração. Em virtude de maior rigor de controle de qualidade durante o projeto, menos erros de projeto permearão para as fases de implementação, teste, integração, conversão e uso. Consequentemente, será significativamente menor o esforço de identificação e remoção das falhas contidas nos programas. Será menor, também, o esforço consumido em reteste do programa.

2- aumento da qualidade do projeto

- o controle automatizado da qualidade, mesmo se incompleto, reduz significativamente o volume de erros que passam despercebidos de uma etapa do desenvolvimento para outra. Em particular, reduz o número de erros que somente serão percebidos durante o uso produtivo do programa.

- a verificação utilizando "animação de projeto" torna mais fácil a uma pessoa não familiarizada com o projeto a entender o que ocorrerá se o programa for implementado tal como projetado. Consequentemente, esta pessoa estará melhor preparada para criticar construtivamente o projeto, contribuindo, assim, para um significativo aumento da qualidade do serviço do programa. Entende-se por qualidade de serviço de um programa a correção, utilidade, utilizabilidade, monitorabilidade, evolutibilidade e economicidade do programa.

- as linguagens de representação utilizadas pelo ambiente padronizam os documentos gerados. Estes padrões podem ser ajustados de modo a tornar os documentos uniformes legíveis, inteligíveis, reprodutíveis e bem acabados. Os formatadores contribuem assim para uma maior qualidade de apresentação da documentação de projeto.

3- redução de danos

- a melhoria de qualidade resulta em economia em função de um menor volume de danos provocados pelo mau funcionamento do programa. São exemplos de danos: prejuízos financeiros, oportunidades perdidas, perda de controle sobre o processo, etc.

- a maior facilidade de manutenção facilita o ajuste dos programas á medida que vão sendo identificadas agressões. Desta forma pode-se tornar mais robusto o programa mesmo que esta robustez não tenha sido previamente projetada.

Referências bibliográficas

DeMarco, T.
Structured analysis and system specification; Prentice Hall; Englewood Cliffs; 1979

Fairley, R.
Software engineering concepts; McGraw Hill; New York; 1985

Freeman, P.; Staa, A.v.
Towards a theory of software engineering; Technical Report TR-242; Information and Computer Science; University of California Irvine; 1984

Freeman, P.; Wasserman, A.I.; eds.
Tutorial on software design techniques; fourth edition; IEEE Computer Society; EH0205-5; New York; 1983

Gane, ; Sarson,
Análise estruturada para sistemas; Livros Técnicos e Científicos; Rio de Janeiro; 1983

Goldberg, A.; Robson, D.
Smalltalk-80 The language and its implementation; Addison Wesley; Reading; 1983

Jackson, M.
Principles of program design; Academic Press; London; 1975

Page-Jones, L.
The practical guide to structured system design; Prentice Hall; Englewood Cliffs; 1980

Staa, A.v.; Freeman, P.
Requirements for software engineering languages; Technical Report TR-85-08; Information and Computer Science; University of California Irvine; 1985

Staa, A.v.
Projeto MOSAICO - Definição do projeto; Departamento de Informática, PUC/RJ; Rio de Janeiro; 1985

Staa, A.v.
Linguagem de representação: Estrutura Organizacional;

Departamento de Informàtica, PUC/RJ; Rio de Janeiro; 1985, em preparação

Staa, A.v.
Projeto e teste de programas estruturados; SCI; Rio de Janeiro; 1985; notas de curso de treinamento

Wirth, N.
Programming in MODULA-2; Springer; Berlin; 1982

Yourdon, E.; Constantine, L.
Structured design; Prentice Hall; Englewood Cliffs; 1979