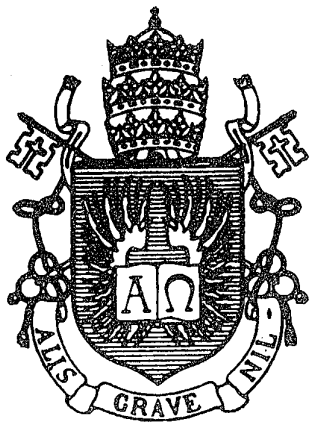


PUC

Série: Monografias em Ciência da Computação

Nº 7/85

PROJETO MOSAICO
DEFINIÇÃO DO PROJETO

por

Arndt von Staa

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

MARQUÊS DE SÃO VICENTE, 225 - CEP-22453

RIO DE JANEIRO - BRASIL

PUC/RJ - DEPARTAMENTO DE INFORMÁTICA

Série: Monografias em Ciência da Computação, Nº 7/85

Editor: Paulo A.S. Veloso Novembro , 1985

PROJETO MOSAICO
DEFINIÇÃO DO PROJETO*

por

Arndt von Staa

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

Resumo

MOSAICO visa apoiar todo o ciclo de vida de software, inclusive a manutenção e o desenvolvimento de eventuais protótipos. MOSAICO tornará disponíveis diversas ferramentas de definição, de edição, de formatação, de controle de qualidade e de transformação das informações contidas em documentos produzidos durante o desenvolvimento e durante a manutenção de sistemas de programação. Cada um destes documentos tem um objetivo informacional específico e destina-se a determinada platéia. Tendo em vista a variedade de sistemas que MOSAICO deverá auxiliar a desenvolver, é objetivo de MOSAICO permitir a especificação e a modificação das especificações dos documentos utilizados.

Uma das características marcantes de MOSAICO é a sua capacidade de transformar informação contida em um documento para informação equivalente contida em outro documento. Esta transformação poderaá ser manual, amparada pelo usuário de MOSAICO, e/ou automática, amparada por sistemas especialistas. Desta forma MOSAICO apoiará todo o ciclo de vida de software desde a especificação até a produção de código executável, incluindo aí também a manutenção.

Todos os componentes de MOSAICO gravitam em torno de uma base de software. A base de software é um banco de dados capaz de armazenar todos fatos que ocorrem em diferentes documentos dos sistemas desenvolvidos com o apoio de MOSAICO. A base de software armazena, também, os descritores de formato dos documentos, os descritores de linguagens de comando e os descritores da própria base de software. Desta forma MOSAICO procura assegurar a adaptabilidade ás diferentes técnicas de desenvolvimento de software, bem como ás diferentes naturezas dos sistemas a serem desenvolvidos com o apoio de MOSAICO.

No presente documento são definidos os objetivos a longo prazo do programa de pesquisa associado ao projeto MOSAICO atualmente em andamento na PUC/RJ. A título de ilustração, são apresentados esboços das organizações funcional e operacional de MOSAICO. Finalmente, é descrita a estratégia de pesquisa e desenvolvimento a ser adotada pelo projeto MOSAICO.

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 é 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 auxílio destas ferramentas.

Durante o desenvolvimento de sistemas de programação são geradas diversas representações. Na maioria das vezes, representações são documentos, porêm podem existir representações que não são documentos no sentido estrito. São exemplos de representações: os documentos externos destinados aos usuários do sistema de programação, os documentos internos destinados ao pessoal técnico, a base de software que armazena as informações de projeto, os textos e diagramas apresentados em telas de computador, etc. Cada uma destas representações tem um objetivo informacional específico e destina-se a determinada platèia, isto è, determina um formato, um ponto de vista e um nível de abstração.

E frequente ser difícil obter um perfeito entendimento do problema a resolver. Consequentemente, é comum efetuar alterações mesmo enquanto o sistema de programação ainda estiver sendo desenvolvido. Estas alterações provocam modificações nas diversas representações. A alteração manual de representações è tediosa e, quando feita, tende a ser incompleta. O resultado è um sèrio desajuste entre o que està implementado e o que se encontra documentado. Este desajuste è amplificado pela manutenção feita de forma indisciplinada.

Para tornar o processo de desenvolvimento de software menos sujeito a panes, têm sido propostas diversas ferramentas de trabalho. Estas ferramentas destinam-se a auxiliar a especificação, o projeto, a construção e o teste de sistemas de programação. Reconheceu-se que ferramentas manuais não são suficientemente eficazes por dependerem demasiadamente da disciplina com que são utilizadas, bem como por gerarem um volume muito elevado de trabalho repetitivo. Consequentemente, a tendência atual è desen-

volver ferramentas automatizadas. Tais ferramentas são capazes de tomarem a si diversas tarefas mecanizàveis do processo de desenvolvimento.

Para serem eficazes, ferramentas devem formar um todo integro, 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 disponiveis hoje. Sem estes objetivos essenciais em mente, corre-se o risco de desenvolver ferramentas caras, supostamente poderosas e que, infelizmente, não trazem os benefícios esperados.

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.

Será definido neste artigo o ambiente de desenvolvimento de sistemas de programação MOSAICO. Este sistema está sendo projetado e desenvolvido na PUC/RJ. A estrutura e a funcionalidade de MOSAICO refletem o modelo de sistema de engenharia de software descrito em [Freeman84, Staa84]. MOSAICO parte do princípio que o desenvolvimento de software è realizado atravès de transformações sucessivas, produzindo e/ou alterando representações.

O ambiente de desenvolvimento MOSAICO manterà uma base de software capaz de armazenar todos fatos que ocorrem em diferentes representações. MOSAICO tornarà disponiveis diversas ferramentas de definição, de edição, de formatação, de controle de qualidade e de transformação destas informações.

Apesar de MOSAICO girar em torno da criação e da manutenção de representações, ele não è um mero sistema de apoio à documentação. A essência de MOSAICO è a transformação de representações, quer seja com o auxilio de um usuàrio, quer seja de modo automàtico. MOSAICO apoiarà a transformação de informação de um formato

de apresentação para outro, sem, no entanto, exigir que informação já existente tenha que ser reintroduzida pelo usuário e, ainda, sem perder ou adulterar informação. Durante o processo de transformação, alguma informação poderá ser inferida a partir de outra informação já conhecida e/ou a partir de conhecimento adquirido em outros projetos de desenvolvimento de software.

2. Caracterização do processo de desenvolvimento de software

A seguir será resumidamente descrito o processo de desenvolvimento e manutenção de sistemas de programação. O objetivo desta descrição é introduzir conceitos básicos de engenharia de software para poder identificar os requisitos que MOSAICO deve satisfazer a fim de tornar-se uma ferramenta eficaz, eficiente e capaz de evoluir no futuro. A presente decrição è um resumo de [Freeman84, Staa85].

2.1 Representações

Nesta seção serão caracterizadas representações e linguagens de representação.

O desenvolvimento racional de sistemas de programação leva á geração de diversos documentos - representações -, tais como especificações, projetos, roteiros de teste etc. Ou seja, o desenvolvimento racional de software envolve a criação e manutenção de diversas representações deste sistema.

Representações podem ser documentos descritivos do software registrados em papel, como, por exemplo: especificações, projetos, listagens de programas fonte, manuais externos etc. Representações podem ser, também, registros armazenados em algum meio computacional, como, por exemplo: bases de software, bibliotecas de programas, bases de dados etc. Finalmente, representações podem ser "documentos" registrados em suportes efêmeros, como, por exemplo: telas de terminal, mensagens em linhas de comunicação etc.

Representações registram (documentam) aspectos do sistema a ser desenvolvido. Cada uma destas representações está redigida em alguma linguagem de representação. Estas linguagens podem ter vários graus de formalização. São exemplos de linguagens de representação: português irrestrito, formulários a serem peenchidos em português irrestrito, diagramas de fluxo de dados, dicionários de dados, diagramas de estrutura organizacional, desenho técnico, linguagem de programação etc.

O objetivo primário de linguagens de representação è servir de mecanismo de comunicação fiel entre os diversos membros da equipe de desenvolvimento, e/ou entre membros da equipe e os usuários, e/ou entre membros da equipe e o sistema de computação utilizado. Portanto, cada linguagem de representação possui formato, sintaxe e semântica próprias, podendo ser formada por um misto de gráficos, tabelas, formulários e texto.

Cada uma das etapas de desenvolvimento de sistemas de programação possui requisitos de comunicação próprios. Consequentemente, cada uma destas etapas requer uma linguagem de representação apropriada aos requisitos de comunicação da etapa. Estes requisitos dependem do grau de abstração, do problema a ser resolvido, do redator e do leitor das representações geradas utilizando esta linguagem.

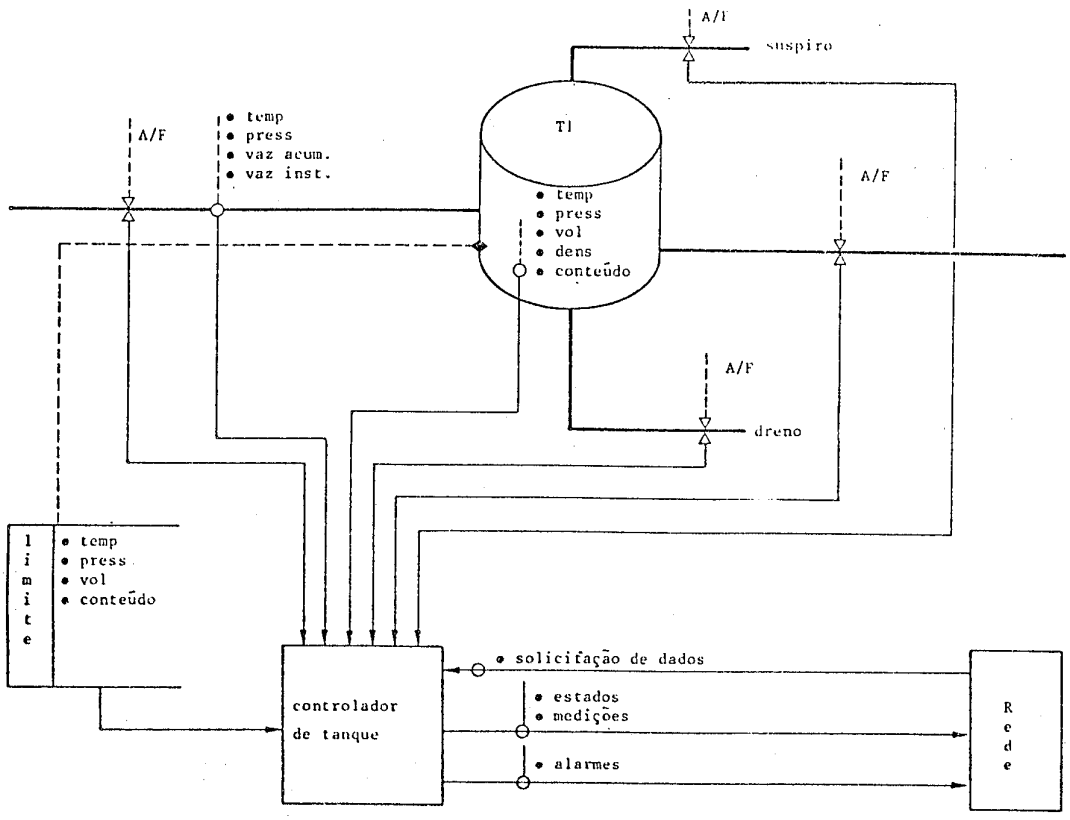


Figura 2.1 Ilustração da linguagem "arquitetura de controle". Esta representação identifica todos os pontos de interface entre um sistema automatizado de controle e o processo (tancagem) controlado por este sistema.

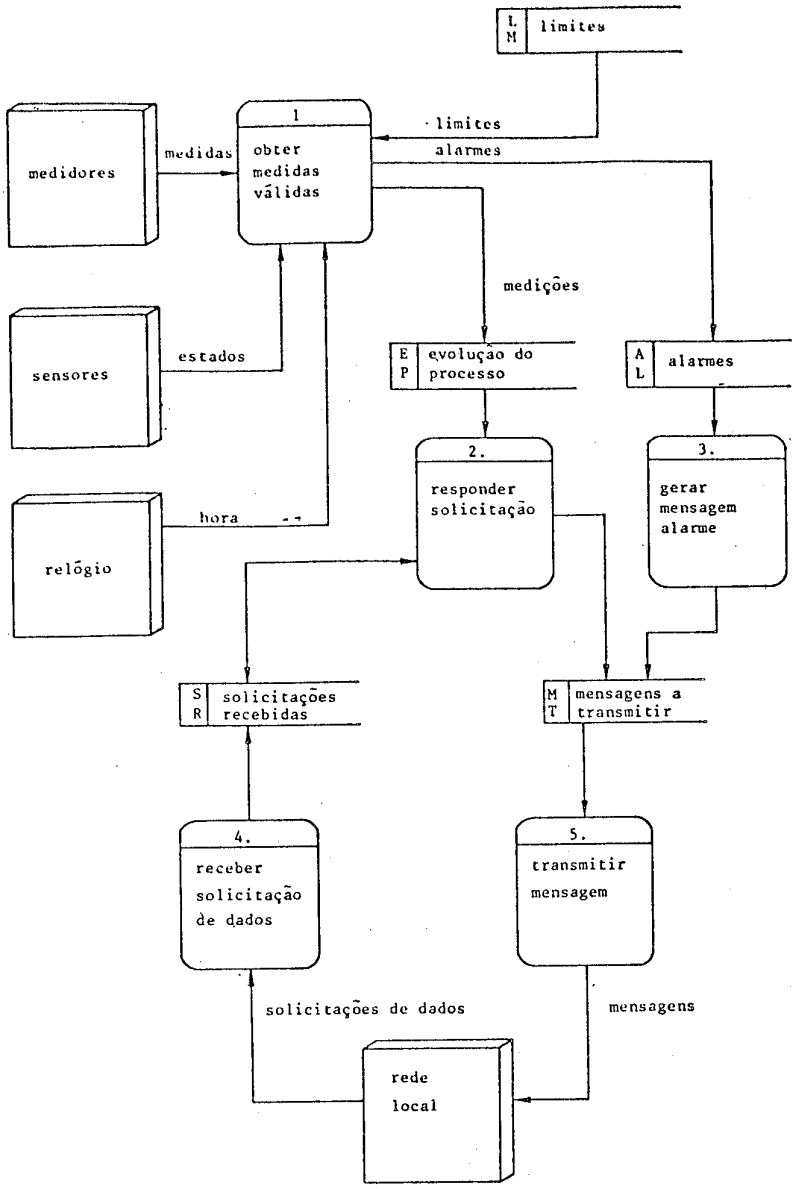


Figura 2.2 Ilustração da linguagem "fluxo de dados".

Nas figuras 2.1 a 2.4 são ilustradas diversas linguagens de representação. Cabe observar que os exemplos são simples e parciais, não formando um conjunto completo e consistente. O objetivo é somente ilustrar algumas das linguagens mais comuns.

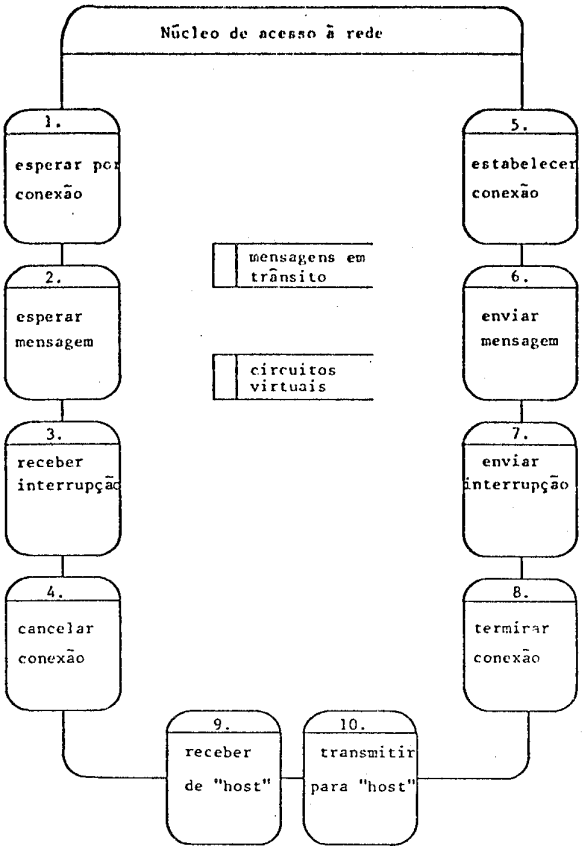


Figura 2.3 Ilustração da linguagem "estrutura funcional".

2.2 Requisitos de representações

Representações não são meros repositórios de informação. Elas são parte ativa do processo de desenvolvimento de sistemas de programação, ver figura 2.5. Para tal, representações precisam satisfazer uma série de requisitos sem os quais serão de pouca serventia. Tais requisitos são:

1- representações devem apoiar a transformação da informação nelas contida. Ou seja, deve ser sempre possível transformar representações em outras, seja através do acréscimo de detalhe (elaboração), seja através da composição de informação contida em várias representações, seja através da abstração ou outra operação qualquer.

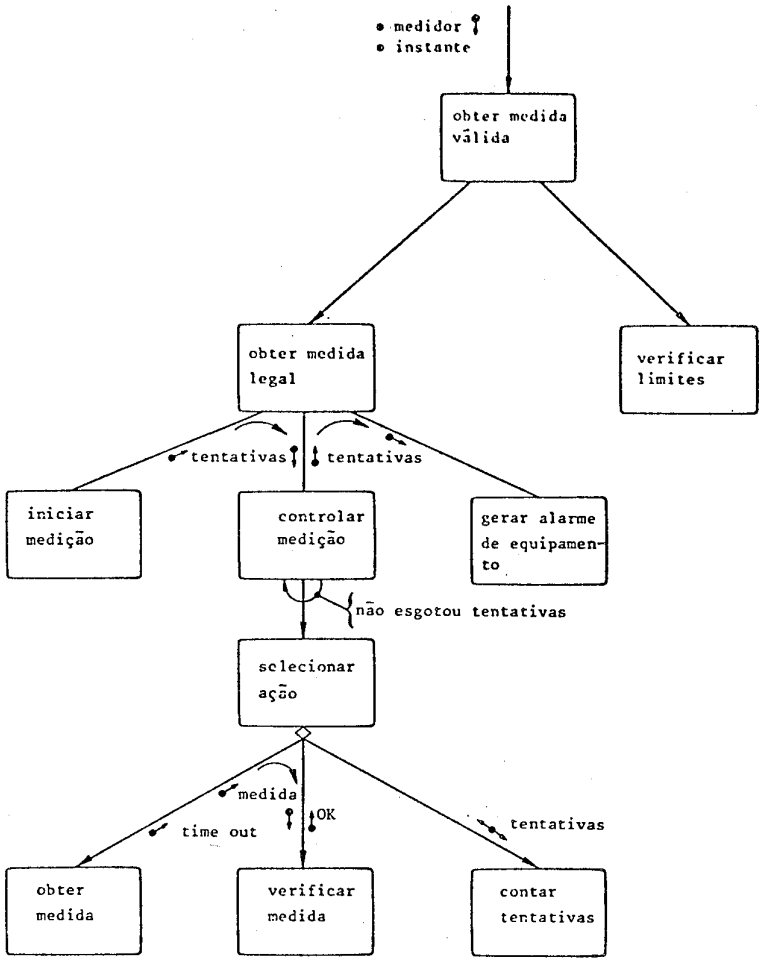


Figura 2.4 Ilustração da linguagem "estrutura organizacional".

Ao transformar representações, diversas operações são efetuadas, ver figura 2.6. O processo de transformação deve poder ser realizado a partir da especificação de requisitos até dispor-se do código e dos dados para teste e conversão. Portanto, durante o processo de desenvolvimento serão criados novas representações, a grande maioria delas em função da derivação a partir de outra representação já existente. No caso extremo, a transformação será realizada adicionando informação externa. Ou seja, a evolução de uma etapa do processo de especificação e/ou projeto é muitas vezes criativa, tornando necessário registrar conhecimento adquirido e/ou decisões tomadas durante a evolução da etapa.

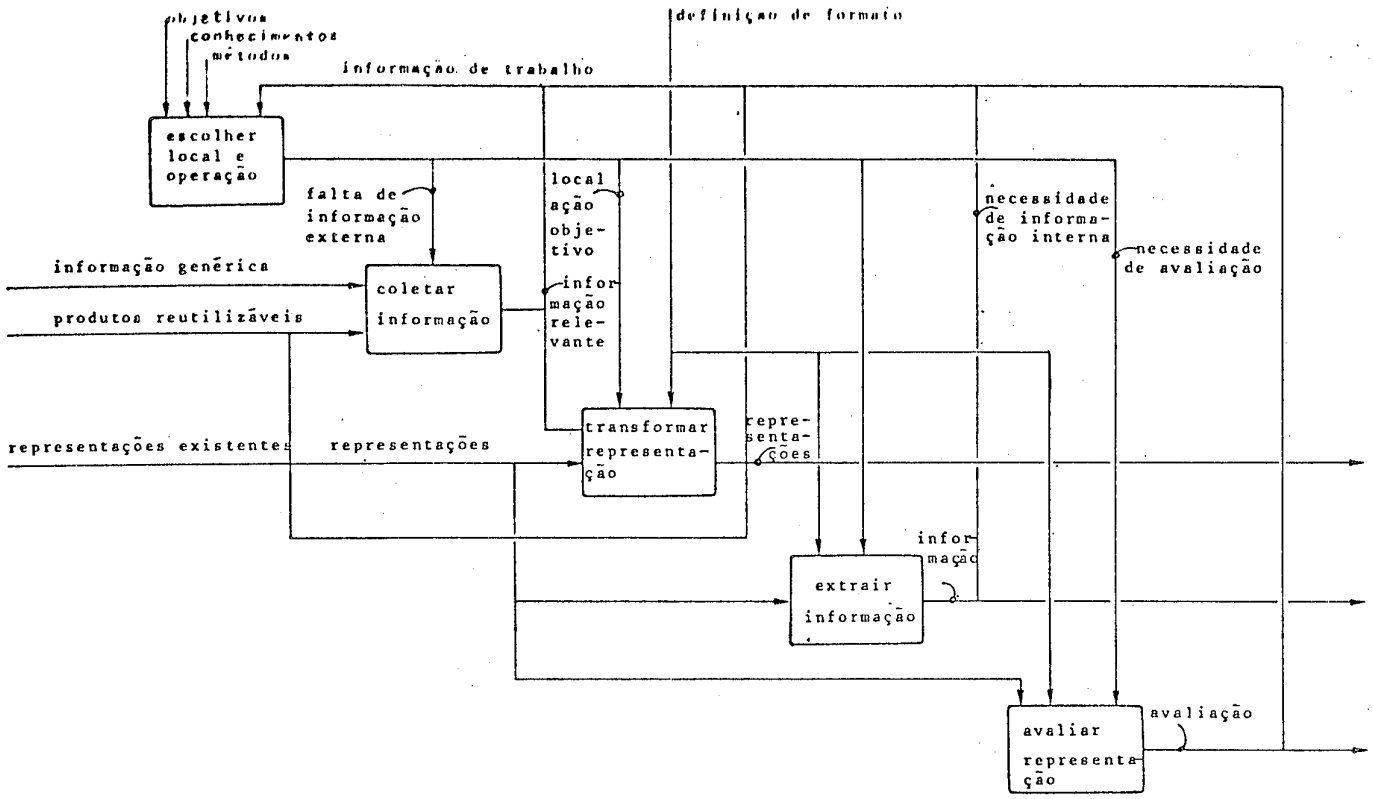
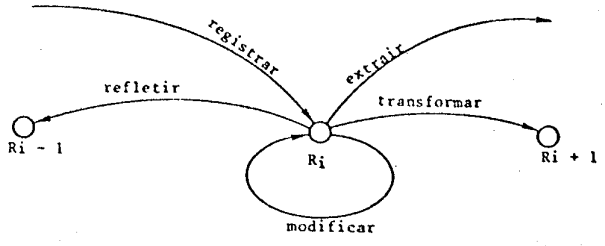


Figura 2.5 O modelo do processo de desenvolvimento e manutenção de software.

Durante o desenvolvimento e, certamente, durante a manutenção, representações poderão ser alteradas. Alterações podem ter por fim corrigir ou aprimorar a representação, tanto no que diz respeito ao seu formato como no que diz respeito ao seu conteúdo.

Alterações podem requerer também a modificação conseqüente de diversas outras representações. É comum que as alterações principiem em representações de baixo nível de abstração (ex. programas), sendo necessário refletir estas alterações em representações de nível de abstração mais alto (ex. especificações). Assim torna-se necessário, também, a possibilidade de iniciar o processo de transformação em representações terminais e refletir as alterações realizadas progressivamente em representações antecessoras, possivelmen-



R_i - representação da i-ésima etapa

Figura 2.6 Operações de transformação sobre representações.

te até alcançar os representações iniciais. Num caso extremo pode-se até desejar reconstruir as representações de projeto e especificação a partir de programas existentes.

O conjunto de representações forma uma rede onde cada nó é uma representação e cada aresta é uma via de transformação direta, ver figura 2.7. Para que seja possível criar e manter esta rede, é essencial que existam caminhos que permitam atingir qualquer representação a partir de qualquer outra representação através de transformações sucessivas. Cada arco destes caminhos define uma linguagem de representação inicial, uma linguagem de representação resultante e regras de transformação.

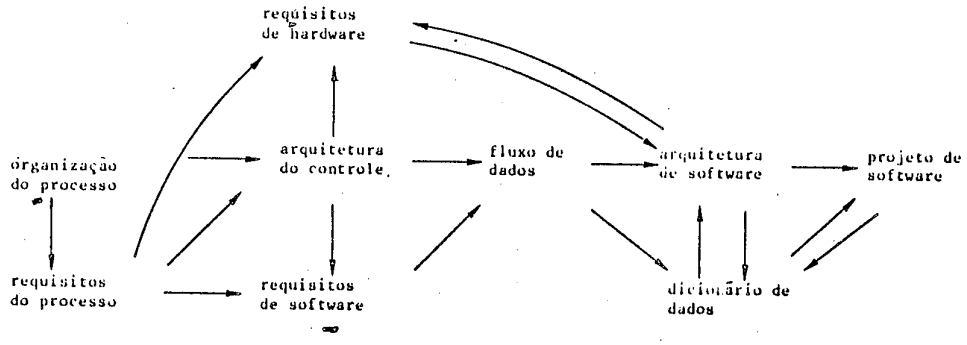


Figura 2.7 Ilustração de uma rede de representações (parcial).

2- representações devem apoiar a comunicação entre pessoas e/ou pessoas e equipamentos. Ou seja, deve ser sempre possível encontrar e extrair a informação procurada a partir do conteúdo da representação.

A extração da informação contida numa representação é fortemente dependente do leitor. Consequentemente, deve ser possível apresentar a mesma informação em diversos formatos e modos de agregação. Cada um destes formatos é dirigido a uma categoria de leitores. Desta forma facilita-se ao leitor obter um perfeito entendimento da informação contida na representação. Como consequência é razoável esperar que o leitor passe a apontar erros, omissões, inadequações etc., contribuindo, assim, para uma melhoria substancial da qualidade do conteúdo da representação. Consequentemente, a revisão da representação efetivamente passa a contribuir para uma melhoria substancial da qualidade do sistema de programação.

Em adição, deve existir uma vinculação entre a linguagem de repesentação externa (papel), interna (tela) e a linguagem de comando (comandos de teclado). Esta vinculação facilita a edição da base de software a partir de alterações anotadas em documentos externos. O formato, o modo (gráfico ou texto) e o detalhe da informação apresentada são vitais para tornar viável e fiel a comunicação homem/sistema.

3- representações devem apoiar o controle de qualidade. Ou seja, á medida que vão sendo geradas e/ou modificadas, as representações deverão ter sua qualidade aprovada.

As representações devem ser examinadas quanto ao correto uso da linguagem de representação e das normas técnicas em vigor, quanto á sua capacidade de comunicação etc. (verificação).

As representações deverão ser examinadas, também, quanto á consistência com outras representações, quanto á correção da derivação a partir de outras representações etc. (validação).

Finalmente, as representações devem ser capazes de indicar se o comportamento do sistema, tal como descrito nestas

representações, tem, ou poderá vir a ter, o nível de qualidade desejado. Deve ser capaz, ainda, de indicar se atende, ou poderá vir a atender, os anseios do usuário do sistema de programação (aprovação). Observe que, para poder efetuar a aprovação, é necessário antes obter-se um perfeito entendimento da informação contida na representação.

4- representações devem apoiar a aquisição de informação relevante. Ou seja, a linguagem de representação utilizada deve dirigir o redator de modo que não omita ou esqueça informação essencial para o correto desenvolvimento do sistema. Entretanto, este direcionamento não deve interferir e/ou dificultar o processo de criação e alteração da representação redigida nesta linguagem de representação.

5- representações devem apoiar a identificação dos pontos de transformação. Ou seja, ao obter informação nova, ou ao obter modificação de informação já conhecida, deve ser possível determinar todas as representações e todos os pontos nestas representações afetados pelo acréscimo ou pela alteração.

Como já foi mencionado, o conjunto de representações forma uma rede dirigida (ver figura 2.7). Cada nó desta rede é uma representação. Cada aresta é uma transformação ou reflexão. Esta rede é criada durante o desenvolvimento. Consequentemente, a precedência é determinada pela sucessão de geração das diversas representações. Usualmente esta sequência prossegue do geral (mais abstrato) para o detalhe (mais elaborado). No entanto, para viabilizar a manutenção, é essencial poder-se também progredir do mais detalhado para o mais geral, portanto em ordem inversa á do desenvolvimento. Um ambiente de desenvolvimento deverá, pois, suportar estas diferentes formas de transformar representações em outras, desde a idéia original até o código de módulos e vice-versa. Cabe observar que, sendo capaz de transformar representações em outras, é possível, também, caminhar sobre a rede. Este caminhamento permite acompanhar a evolução dos requisitos nesta rede. Isto é fundamental para o controle da qualidade (rastreamento de requisitos), bem como para a manutenção (avaliação do impacto de alteração).

2.3 Desenvolvimento e manutenção de sistemas de programação

Nesta seção será examinado, brevemente, o processo de desenvolvimento e manutenção de sistemas de programação como um todo. Ao leitor interessado em maiores detalhes sugerem-se as seguintes referências [Fairley85, Pressman82].

Ao utilizar preceitos da engenharia de software o desenvolvimento de sistemas de programação evolui do geral para o detalhe. Na figura 2.8 são ilustradas as etapas do processo de desenvolvimento de sistemas de programação (ciclo de vida).

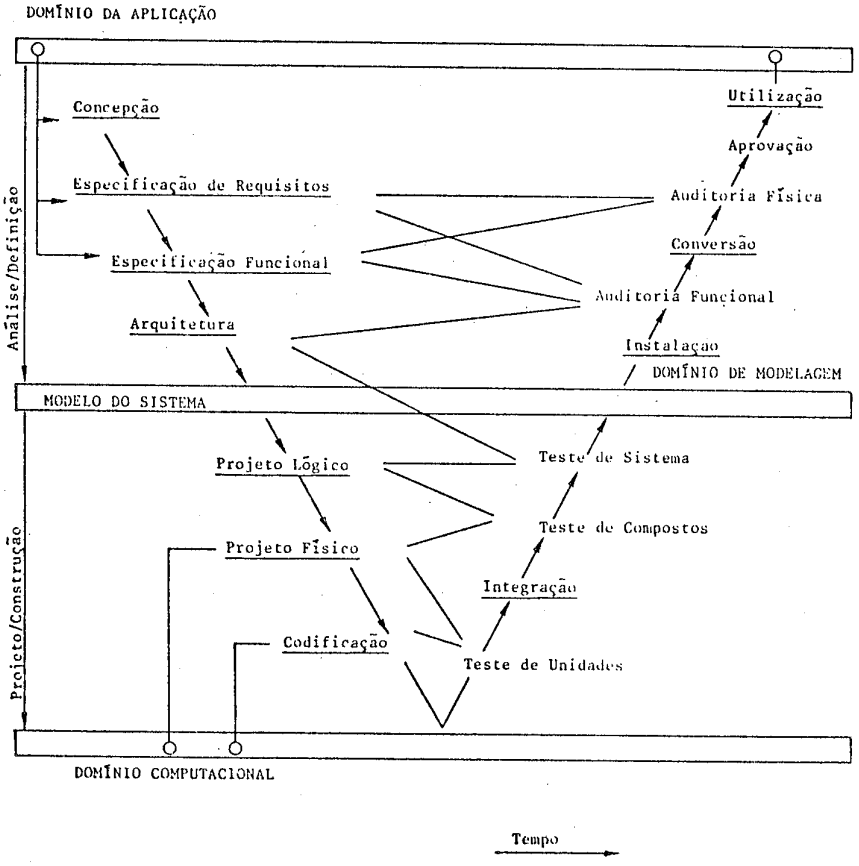


Figura 2.8. O ciclo de desenvolvimento.

O desenvolvimento começa com uma idéia do sistema de programação a desenvolver. A seguir são definidos os requisitos a serem satisfeitos por este sistema. Durante a etapa de definição dos requisitos podem ser conduzidos experimentos baseados em simulações e/ou protótipos. É importante, pois, que um ambiente de

desenvolvimento permita a ràpida construção de modelos e de protòtipos. E importante, tambèm, que este ambiente facilite a incorporação dos conhecimentos adquiridos nas especificações do produto a ser efetivamente desenvolvido (ver figura 2.9). Uma vez de posse da especificação de requisitos, è projetada uma solução que satisfaça estes requisitos. Após è implementada esta solução. A implementação se dá passo a passo, desenvolvendo componentes, integrando estes componentes formando compostos, atè que, eventualmente, o sistema esteja totalmente desenvolvido. Finalmente, è efetuado um controle da qualidade operacional do sistema tal como implementado.

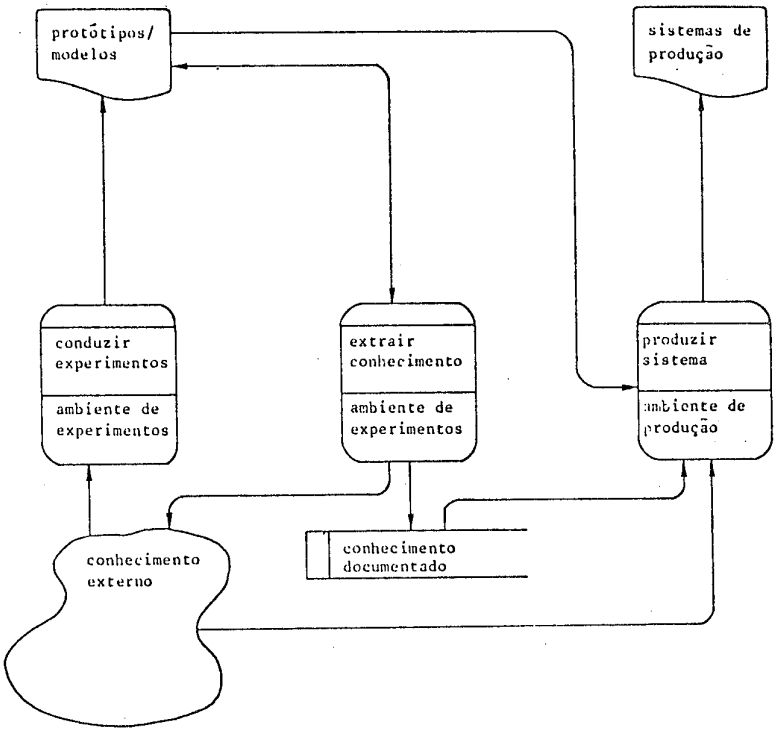


Figura 2.9 Interação entre os ambientes de experimentação e de desenvolvimento.

Uma vez desenvolvido o sistema, inicia-se a sua manutenção. Esta è essencialmente um sucessão de alterações efetuadas no sistema. Vàrias podem ser as causas das alterações no sistema, por exemplo: acrescentar novas funções; adaptar o sistema a novas condições de funcionamento; melhorar o desempenho; corrigir defeitos; etc. Cabe observar que tarefas de manutenção maiores

usualmente são realizadas sob a forma de projetos de modificação do sistema. Cabe observar, ainda, que já durante o próprio processo de desenvolvimento poderão ser realizadas alterações (manutenção durante desenvolvimento).

Para assegurar vida longa ao sistema, é necessário que a manutenção não reduza o nível de qualidade do sistema. É pois, imprescindível que sejam alterados não só os programas, mas, sim, programas e toda a documentação associada. Consequentemente, os instrumentos utilizados durante o desenvolvimento devem poder ser utilizados também durante a manutenção. Além disto, o custo da manutenção deve ser, no máximo, proporcional ao volume das alterações. Para tal é necessário poder-se restringir o escopo de uma alteração a exatamente os pontos alterados.

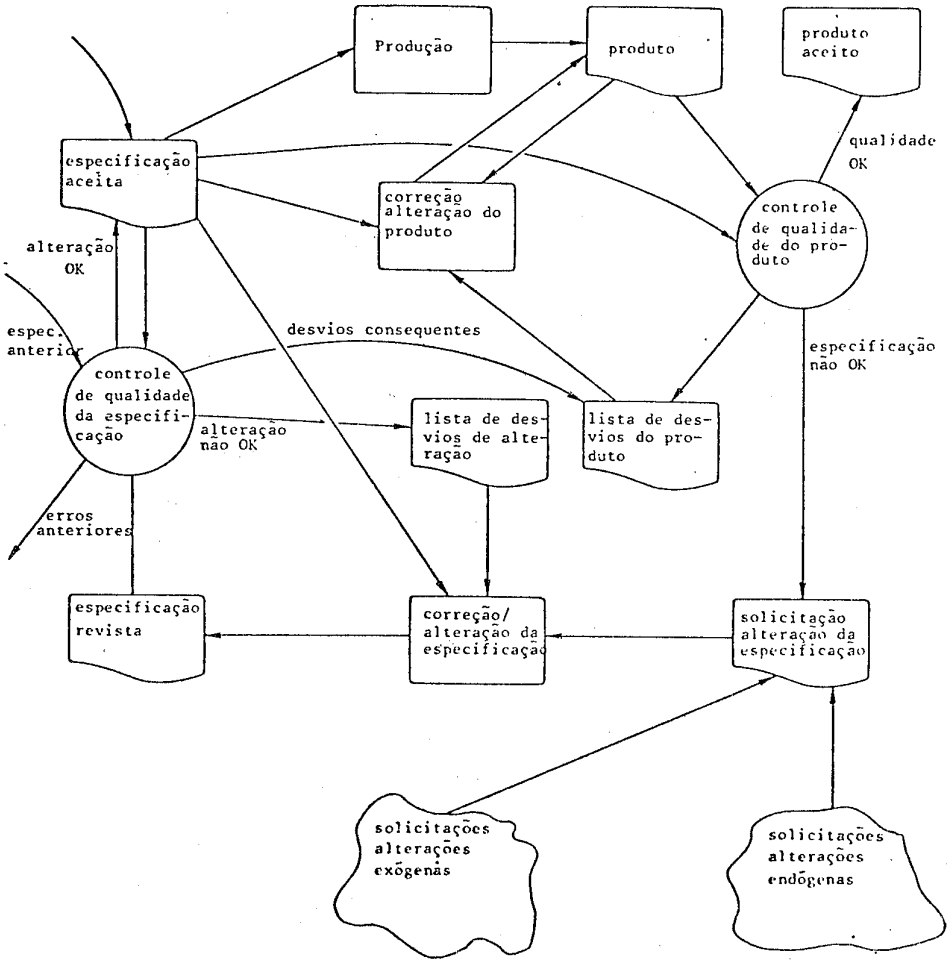
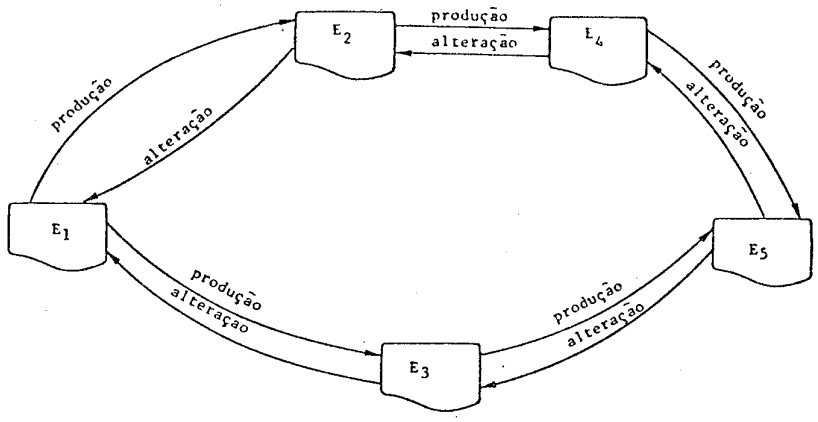
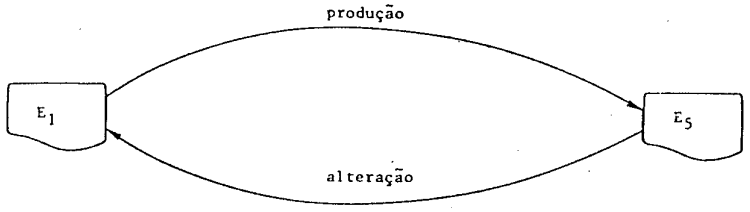


Figura 2.10 O ciclo de produção.

Durante o desenvolvimento a qualidade de especificações e de segmentos de código é constantemente controlada. O controle de qualidade visa assegurar que se esteja desenvolvendo corretamente o sistema correto. Pode-se visualizar o controle de qualidade como sendo a atividade conclusiva do ciclo de produção de uma representação (ver figura 2.10). O desenvolvimento é, pois, uma sequência destes ciclos de produção (ver figura 2.11).



sequência de produção detalhada



sequência de produção agregada (hierarquia)

Figura 2.11 Composição e hierarquia de ciclos de produção.

O controle de qualidade será frequentemente efetuado a partir de especificações ou a partir de projetos. É, pois, essencial que linguagens de representação permitam realizar este controle de qualidade. Para tal é necessário que as representações sejam legíveis e inteligíveis pelos diversos leitores, leigos ou não em computação.

As etapas de controle de qualidade são:

1- verificação - examinar a qualidade da representação em si. Ou seja, examinar se a representação está corretamente redigida, se obedece ás normas tècnicas e legais em vigor, se è legìvel e inteligìvel, etc.

2- validação - examinar a qualidade do conjunto de representações geradas atê o momento. Ou seja, examinar se a representação corresponde a uma correta transformação ou reflexão de uma outra representação, verificar se o conjunto de representações è consistente, etc.

3- aprovação - examinar a qualidade do serviço do sistema ou componente a que se refere a representação. Ou seja, examinar se o sistema ou componente corresponde ás necessidades e aos desejos do usuàrio. A aprovação poderà ser preditiva, por exemplo ao examinar uma especificação com o intuito de assegurar que o sistema se comportarà conforme esperado pelo usuàrio. A aprovação poderà ser efetiva, por exemplo ao testar um componente do sistema, ou ao conduzir experimentos utilizando um protòtipo.

2.4 Problemas do processo de desenvolvimento tradicional

A criação e a manutenção de representações è usualmente efetuada de forma manual. E despendido muito esforço para produzir e para alterar estes documentos. Este esforço è particularmente elevado caso se deseje que o resultado final seja confiàvel e consistente e esteja em formato aceitàvel, reprodutìvel e legìvel. E exatamente isto que se deseja ter ao final do desenvolvimento de software de boa qualidade.

Jà foi mencionado que o desenvolvimento e a alteração de um sistema de programação se dà atravès de uma sèrie transformações, cada uma resultando em uma representação contida na rede de representações. A transformação de representações redigidas numa linguagem de representação para representações redigidas em outra linguagem tambèm tem sido efetuada de modo manual.

No conjunto de representações da rede de representações

existe muita redundância, uma vez que um mesmo fato poderá ser definido em diferentes representações, utilizando somente diferentes formas de apresentação (linguagens de representação). Estas diferentes formas de apresentação são necessárias de modo que os representações possam satisfazer os requisitos essenciais acima mencionados, em particular para servirem de veículos de comunicação entre pessoas.

É óbvio que a criação e a manutenção manual da rede de representações tenderá a resultar em erros de registro, em erros de transformação, em inconsistência entre as diversa representações geradas etc. A consequência final disto é a perda de visibilidade e de controle do processo de desenvolvimento e manutenção.

Como tentativa de sanar estes problemas têm sido propostas diversas medidas paliativas. São medidas paliativas uma vez que não atacam o âmago da questão, contentando-se a reduzir a intensidade dos sintomas.

Um exemplo típico é o uso de metodologias amparadas em diversas linguagens de representação, possivelmente até mecanizadas. Esta é uma medida paliativa pois:

i) não elimina a repetição de esforço de registro de um mesmo fato em documentos diferentes e

ii) não define com precisão e algumas vezes até impede a transformação de um documento qualquer em outro documento consistente com o primeiro.

Outro exemplo de medida paliativa é a aplicação de medidas gerenciais restritivas amparadas em inúmeras normas. Isto também não resolve, pois o problema de correto registro, extração e transformação de informação não é um mero problema de disciplina, apesar desta ser benéfica no sentido de reduzir o número de problemas enfrentados.

Precisamos pois criar um ambiente de desenvolvimento de sistemas de programação capaz de:

1- lidar com uma rede de representações composta por uma miríade de diferentes representações e linguagens de

representação.

2- apoiar a transformação destas representações de modo que se possa criar e caminhar nesta rede de representações.

3- poupar esforço na criação, manutenção e consulta a estes representações.

Um ambiente de desenvolvimento satisfazendo estes objetivos è essencialmente um sistema de informação. Como tal ele serà capaz de registrar fatos sobre o sistema de programação em questão e serà capaz de apresentar estes fatos em vàrias formas e niveis de agregação.

3. Objetivos do sistema MOSAICO

MOSAICO apoiará o processo de registro das informações que compõem uma representação. Apoiará, também, o processo de transformação de representações em outras. No entanto, MOSAICO não estará vinculado a uma metodologia específica. Assim MOSAICO, em princípio, não imporá restrições de linguagem de representação e de sequência de desenvolvimento. Para poder criar e manter reresentações é claro que determinada linguagem de representação será utilizada, porém é esperado ser pequeno o esforço de criação e inclusão de novas linguagens de representação em MOSAICO.

MOSAICO será um ambiente de desenvolvimento de sistemas de programação. MOSAICO viabilizará a especificação, o projeto, a codificação e o controle de qualidade assistidos por computador.

São objetivos de MOSAICO:

1- prover um sistema de informação técnica - a base de software. A base de software registra, em formato interno, todos os fatos relativos aos diversos sistemas sendo desenvolvidos ou mantidos por intermédio de MOSAICO. A base de software contribui para a eliminação de esforço redundante, pois permite a extração de diferentes informações apresentando-as em diferentes formatos, a partir de um mesmo conjunto de fatos. Além disto, a base de software permite o registro e a alteração de fatos a partir de qualquer documento da rede de documentos que contenha informação sobre estes fatos.

2- prover diversos formatadores capazes de exibir a informação contida na base de software. Esta informação será exibida em formato e grau de detalhe adequado ao uso que se fará do documento. Através dos formatadores é reduzido o esforço de criação, alteração e regeneração de documentos, uma vez que os documentos passarão a ser gerados diretamente a partir dos fatos registrados na base de software.

3- prover diversos editores interativos. Estes editores permitem a criação e/ou alteração da informação contida na base de software. Editores são um misto harmonioso de linguagens de comando, linguagens de representação e atualizadores da base de software. Através do vínculo com as linguagens de repre-

sentação, os editores permitem fechar a malha de criação e manutenção dos fatos relevantes do sistema de programação em questão. Editores interativos reduzem o esforço de desenvolvimento, pois permitem realizar um controle de qualidade parcial já ao registrar ou alterar fatos. Reduzem o esforço também por colocarem o engenheiro de software em uma malha fechada e dinâmica de ação e consequência durante a criação, a manutenção e a consulta á rede de representações.

4- prover mecanismos para o reaproveitamento de projetos anteriores através da instanciação de esquemas e/ou através da modificação sistemática de sistemas já anteriormente desenvolvidos (reutilização generalizada). A reutilização reduz o esforço necessário, uma vez que parte significativa de qualquer desenvolvimento ou alteração nada mais é do que refazer algo que já havia sido feito no passado.

5- prover suporte automatizado ao controle de qualidade (verificação, validação e aprovação) dos diversos componentes desenvolvidos. Em particular é de extremo interesse que o ambiente forneça apoio á aprovação de especificações e de projetos. O controle de qualidade continuado reduz significativamente o esforço perdido em função do desenvolvimento a partir de especificações erradas.

6- prover suporte automatizado á transformação da informação contida na base de software. Um tal suporte é capaz de gerar propostas de solução a partir de regras pré-definidas e/ou a partir de conhecimento adquirido em desenvolvimentos anteriores, através do uso de sistemas especialistas. Este suporte reduz o esforço de desenvolvimento e manutenção, pois automatiza tarefas mecanizáveis, mesmo as que aparentemente são de natureza criativa. No caso extremo, um transformador de informação transformará uma especificação de alto nível de abstração em código executável. Ou, então, executará diretamente a partir da especificação. Um tal transformador de informação é essencialmente um gerador de protótipos.

7- prover suporte ao controle de alterações. Um tal suporte reduz o esforço perdido, pois reduz o volume de ajustes necessário para compatibilizar componentes desenvolvidos por diferentes equipes ou pessoas.

8- prover suporte para a comunicação entre equipes. Através deste suporte é reduzido o esforço despendido para o estabelecimento da comunicação entre equipes. Além disto, torna-se possível manter todas as equipes informadas com as versões mais atualizadas dos diferentes fatos e documentos, reduzindo assim o esforço perdido ao desenvolver a partir de especificações obsoletas.

9- prover suporte mecanizado á gerência de configuração. Através deste suporte é facilitada a integração de sistemas a partir de componentes pré-definidos, assegurando-se a integridade do conjunto mesmo no evento de alterações (manutenção) em um ou mais destes componentes.

É óbvio que os objetivos de MOSAICO acima expostos são muito ambiciosos. Para o estado atual da arte, estes objetivos constituem uma diretriz de pesquisa avançada e, certamente, um projeto de prazos e custos por ora não determináveis. Estes objetivos de MOSAICO servem, pois, como um mecanismo de coordenação e de direcionamento de esforço de pesquisa, onde esta pesquisa está ainda descompromissada com a possível rentabilidade de MOSAICO. O compromisso desta pesquisa é o de demonstrar de modo construtivo e experimental a viabilidade técnica de MOSAICO.

4. Arquitetura funcional de MOSAICO

Nas figuras 4.1 e 4.2 é esboçada uma possível organização funcional do sistema MOSAICO. Este esboço serve exclusivamente como ilustração de como poderiam vir a ser alcançados os objetivos funcionais de MOSAICO.

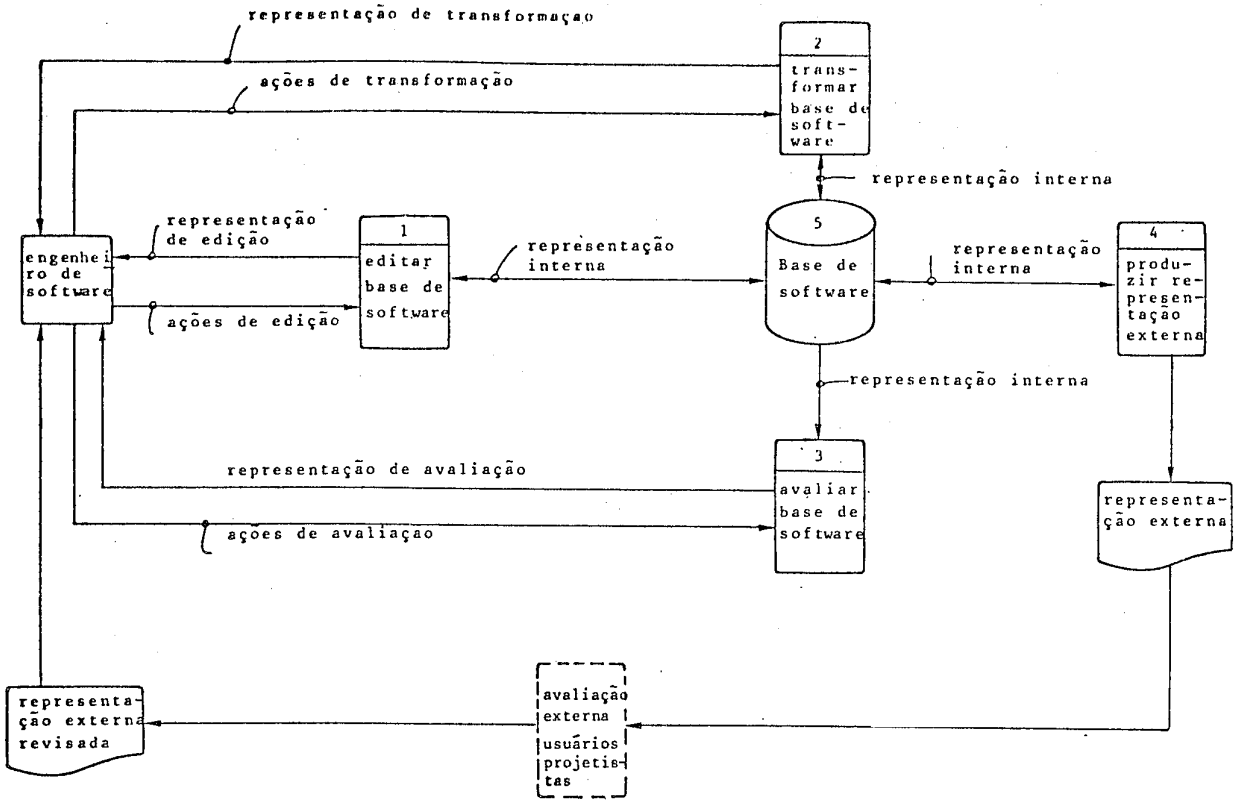


Figura 4.1 Fluxo geral.

A seguir descrevemos em linhas gerais alguns dos componentes do sistema MOSAICO:

1- base de software. Este è um banco de dados projetado para armazenar todos os fatos dos diversos sistemas de programação em desenvolvimento ou manutenção. A base de software deve facilitar o acesso aos diversos fatos, sendo essencial que se possa consultar e atualizar a base de software a partir de qualquer representação. Ou seja, è essencial quë qualquer linguagem de representação possa ser utilizada para atualizar e/ou consultar a base de software.

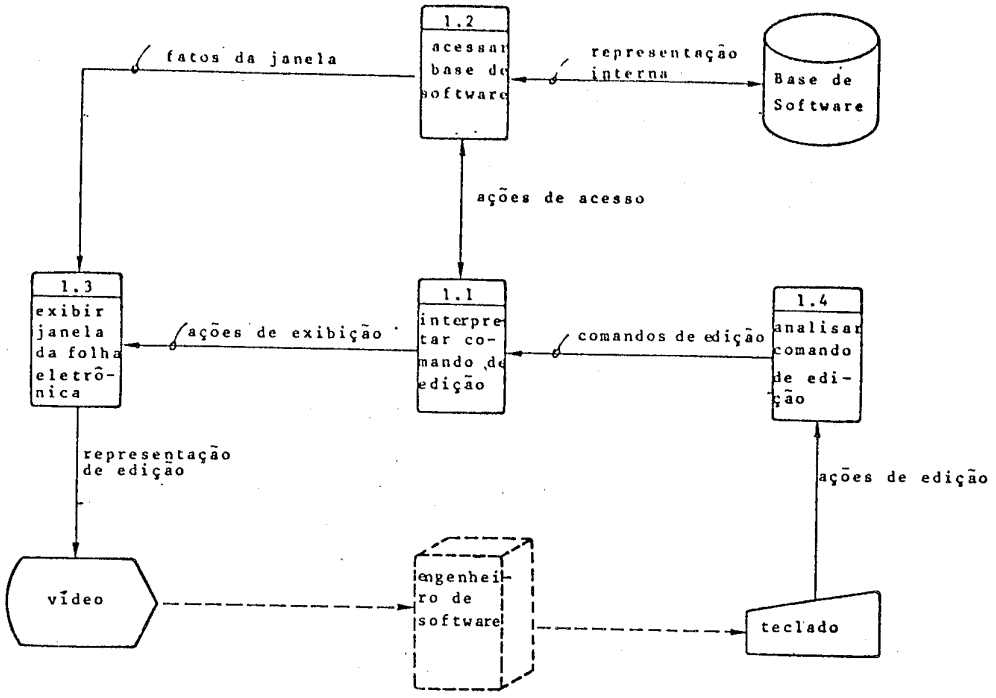


Figura 4.2 Fluxo dos editores.

Consequentemente, a característica essencial da base de software é a variedade de formatos e organizações de dados que deverá suportar. Cabe observar que estas organizações não são especificáveis a priori, sendo, portanto, necessário poder definir a organização da base de software a medida que se vai progredindo no desenvolvimento de MOSAICO e dos sistemas desenvolvidos com suporte de MOSAICO.

2- definidor de linguagem de representação. Qualquer linguagem de representação tem um formato interno definido pela base de software e possui, também, um formato externo utilizado como veículo de comunicação com o usuário de MOSAICO.

O formato externo das linguagens de representação é definido através de formatadores. Estes formatadores são programas capazes de exibir em algum dispositivo de exibição (terminal, impressora, etc.) uma parte selecionada do conteúdo da base de software. A formatação do conteúdo da base de software em algum veículo "hardcopy" viabiliza a criação

de documentos e facilita o exame crítico de documentos sem a necessidade da presença física de equipamento computacional.

Uma parcela considerável de linguagens de representação externas são gráficas. Consequentemente, os formatadores utilizados para gerar as representações externas deverão ser capazes de produzir texto formatado gráfico.

O definidor de linguagens de representação deverá ser capaz de especificar, para cada uma das diversas linguagens de representação utilizadas, o formato, o modo de definir a janela visível no terminal, e a vinculação destes com a base de software.

3- geradores de editores da base de software. Editores são sistemas de registro, consulta e edição de fatos contidos na base de software. Para assegurar elevada produtividade (produtos corretos por unidade de custo) é essencial que os editores possuam excelentes características de comunicação homem/sistema.

Para estabelecerem a interface homem/sistema, os editores precisam exibir, de modo formatado, o conteúdo da base de software. Consequentemente, editores estão associados a formatadores. Estes devem refletir exatamente o conteúdo atual da base de software, onde este conteúdo está, possivelmente, em processo de ser modificado. Em particular, durante a criação da base de software este conteúdo poderá estar ainda indefinido. Tornam-se necessárias, então, convenções para lidar com bases de software possuindo conteúdo indefinido, incorreto e/ou incompleto. Finalmente, cabe salientar que os formatadores poderão variar, dependendo do usuário de MOSAICO, e da natureza do sistema sendo desenvolvido com apoio de MOSAICO.

Geradores de editores são sistemas que associam uma linguagem de comando a um formatador e á organização da base de software. Este conjunto opera de modo interativo e em tempo real. Assegura-se assim que o usuário de MOSAICO esteja inserido em uma malha de ação e consequência necessária para assegurar uma boa interface homem/sistema.

Tendo em vista que uma parcela considerável de comunicação é efetuada a partir de documentos físicos, é necessário, ainda, que a linguagem de representação utilizada pelo editor seja semelhante, ou até igual, à linguagem de representação externa utilizada em documentos físicos.

4- transformadores de informação. Estes são processos que efetuam a transformação do conteúdo da base de software a partir dela mesma. Esta transformação poderá ser realizada de modo automático, e poderá ser, também, realizada com a assistência de um engenheiro de software, diretamente e/ou por meio de um sistema especialista. Através da transformação é possível evoluir em grau de detalhe e/ou instanciar esquemas de especificações e de projetos já desenvolvidos no passado. Além disto, transformadores são responsáveis pela geração de protótipos a partir do conteúdo da base de software. A essência inteligente do ambiente encontra-se nestes transformadores.

5- controladores de qualidade. Controladores de qualidade são ferramentas que inspecionam a base de software com o intuito de avaliar o conteúdo desta base de software segundo diversos critérios de integridade. Os controladores de qualidade poderão estar operando em conjunto com os editores da base de dados. Desta forma torna-se possível controlar critérios de integridade dinâmica da base de software. Controladores de qualidade poderão, também, ser acionados após ter-se chegado a determinado ponto no processo de desenvolvimento. Desta forma os controladores de qualidade poderão ser utilizados como avaliadores do progresso do desenvolvimento. Ou seja, caso a qualidade avaliada em determinado ponto seja aceitável, os controladores de qualidade utilizados confirmam ter-se alcançado determinado ponto de controle do plano de execução.

Os controladores de qualidade podem ser, num extremo, verificadores e validadores formais de porções da base de software. Em outro extremo, podem ser interpretadores capazes de descrever o funcionamento do sistema através de suporte mecanizado ao "structured walkthrough" (animação de projeto).

5. Arquitetura física de MOSAICO

Na figura 5.1 esboçamos, a título de ilustração, uma possível organização física de MOSAICO. Cada usuário de MOSAICO interagirá com uma estação de trabalho. Esta estação é um micro computador. Este micro computador não deve impor restrições de memória, tempo de processamento, capacidade semi-gráfica, etc.

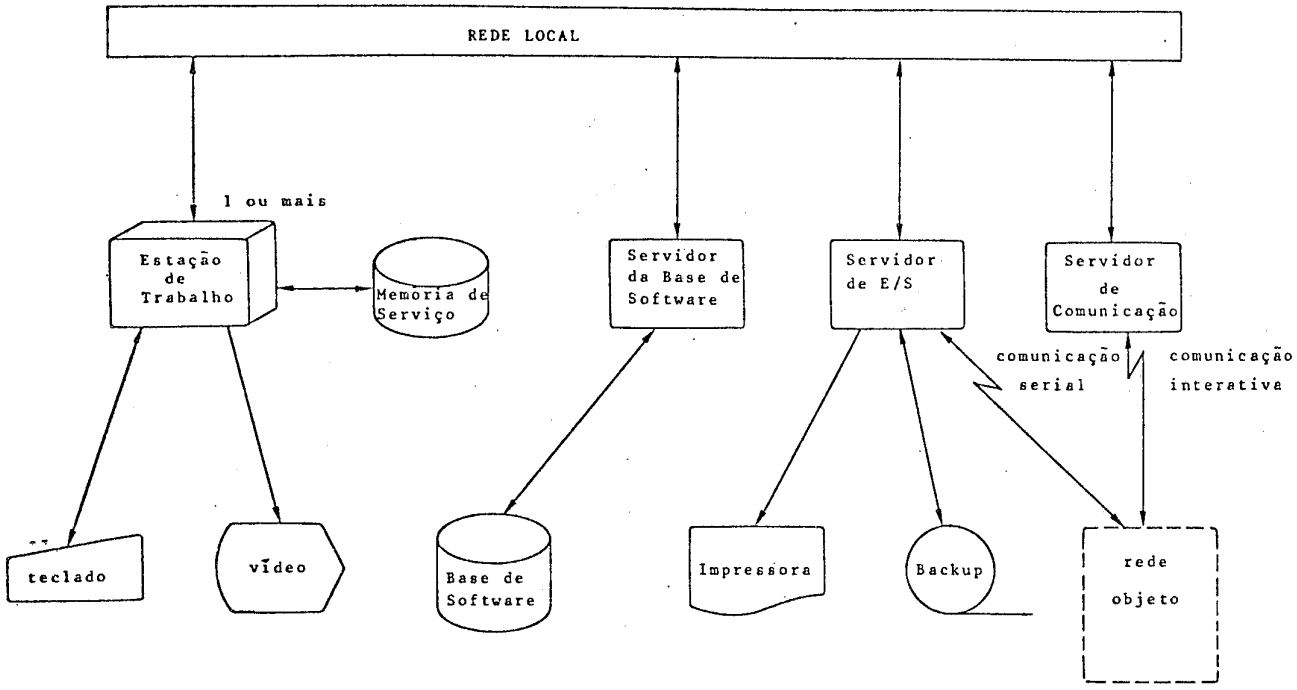


Figura 6. Organização operacional de MOSAICO.

A base de software será gerenciada por um micro computador próprio. Uma das tarefas deste servidor é controlar o acesso múltiplo a um mesmo conjunto de fatos. Cabe salientar que a base de software é temporariamente distribuída. Ou seja, durante o uso das diferentes estações de trabalho, cada uma destas estações possuirá cópias de porções da base de software, tornando possível, assim, a edição destas porções sem induzir demasiadas delongas.

A interação entre estações de trabalho e outros servidores, será realizada através de uma rede local. Para poder utilizar

MOSAICO como ferramenta de desenvolvimento de sistemas que irão eventualmente operar em computadores de grande porte, è necessàrio que a rede local possa interagir com este processador objeto.

Ao utilizar MOSAICO para desenvolver sistemas que operarão em outro computador objeto, è necessàrio que as estações de trabalho possam vir a operar como se fossem terminais deste computador objeto. A comunicação interativa serà gerenciada por um micro computador que tem por finalidade converter protocolos da rede local para os da rede objeto e vice-versa. Tem por missão, ainda, a tradução da linguagem de comando utilizada na rede local para a linguagem de comando utilizada na rede objeto e vice-versa.

Finalmente, tarefas de MOSAICO que não necessitem da supervisão interativa de algum usuàrio, (por exemplo geração de documentos, transmissão de arquivos etc.) poderão ser realizadas por sua vez por servidores implementados em micro computadores. Prevendo a transmissão de arquivos para a rede objeto, torna-se inclusive possível a utilização do computador objeto como uma das ferramentas de suporte a MOSAICO.

6. Benefícios esperados ao utilizar MOSAICO

Usando MOSAICO o desenvolvimento de programas passa a ser uma tarefa de:

1- definição e adaptação de linguagens de representação do problema a ser desenvolvido. Assegura-se assim uma melhor comunicação com usuários possivelmente leigos em computação.

2- emprego destas linguagens de representação para registrar a intenção do engenheiro de software. Cabe salientar que qualquer linguagem poderá ser utilizada para a edição da base de software, sendo que porções comuns a diversas linguagens serão atualizadas "simultaneamente" em tôdas estas representações.

3- instanciação de esquemas de programas, projetos e/ou especificações, gerando, assim, uma versão ajustada ao problema em questão.

4- controle da qualidade das diversas representações, tanto teórica como experimental de especificações, projetos e programs.

MOSAICO é, pois, essencialmente um sistema de "computer aided design" (CAD) projetado para os problemas enfrentados durante o desenvolvimento de sistemas de programação.

MOSAICO reduz substancialmente o esforço de desenvolvimento de software, através da eliminação de redundância e repetição na redação, tradução, e geração de documentos. MOSAICO reduz os erros através de uma maior incidência de controle de qualidade durante o desenvolvimento do sistema de programação, além de viabilizar a condução de experimentos cedo ("fast prototyping").

7. Estratègia para o desenvolvimento de MOSAICO

Como jà foi mencionado, MOSAICO è um projeto ambicioso. Necessita-se, então, de uma estratègia de desenvolvimento de modo que se torne possível o alcance dos objetivos traçados.

Essencialmente seguir-se-à uma estratègia de implementação incremental. Ou seja, serão desenvolvidas ferramentas que por si só já trazem benefícios. Estas ferramentas estarão sempre em consonância com os objetivos gerais traçados. Garante-se assim a possibilidade de integração futura destas diversas ferramentas.

Uma vez completada uma etapa, as ferramentas tornadas disponíveis serão utilizadas para projetar e implementar outras. Desta forma, alèm de auxiliarem no desenvolvimento, as ferramentas desenvolvidas tambèm passarão por um uso experimental, permitindo, assim, avaliar a sua eficàcia.

Em paralelo com o desenvolvimento de ferramentas úteis, serão desenvolvidos trabalhos de investigação científica visando formalizar, fundamentar e/ou avaliar os passos futuros. Desta forma, MOSAICO torna-se uma base para a condução de pesquisa em engenharia de software, alèm de ser uma base para o desenvolvimento econômico e racional de software.

Referências bibliográficas

Obs. A presente lista de referências bibliográficas cita somente os artigos inspiradores do princípio básico de MOSAICO - desenvolvimento através de transformações sucessivas de documentos. Uma revisão da literatura relativa a ambientes de desenvolvimento de software citando e fornecednndo resenhas das diversas referências, será apresentada em documento futuro.

[Freeman 84] Freeman, P.; Staa, A.v.
Towards a theory of software engineering; Technical Report #242; Information and Computer Science; University of California, Irvine; novembro 1984;

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