UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA DE CAMPINAS

DEPARTAMENTO DE ENGENHARIA ELETRICA

Este exemplar corresponde a redação final da tese defendida por Marcelo Vieira Silva e aprovada pela comissão julgadora em 13/5/85

L. P. Magalhães

PROCESSADOR DE DIALOGO : Uma Ferramenta Para Geração de Sistemas Interativos

por Eng. Marcelo Vieira Silva
Orientador: Prof. Dr. Léo Pini Magalhães

027/85

Tese de Mestrado apresentada à Faculdade de Engenharia de Campinas, da Universidade Estadual de Campinas.

MAIO - 1985

Agradeço ao CNPQ - Conselho Nacional do Desenvolvimento Científico e Tecnológico pelo apoio que este trabalho recebeu na forma de bolsa de estudos.

Agradecimentos

- ao Prof. Dr. Léo Pini Magalhães, pela orientação durante todo o Curso de Mestrado e, em especial, para a realização deste trabalho;

- à Prof. Beatriz Mascia Daltrini, pelo apoio em todas as atividades escolares paralelas ao Mestrado;

- ao Prof. Dr. Clésio Luis Tozzi, pelas discussões elucidativas com relação ao campo da computação gráfica;

- ao Eng. Aguinaldo Rodrigues de Campos Júnior, pela colaboração nas atividades computacionais;

- ao Prof. José Raimundo de Oliveira, pelo auxílio na utilização dos equipamentos de laboratório, principalmente o sistema de edição e impressão de textos;

- às famílias de Ailton e Rosa, pela convivência e apoio;

- a minha família, pela paciência e compreensão durante os três anos de atividades em Campinas;

- e, de modo geral, a todos aqueles que, a qualquer tempo, prestaram colaboração a este trabalho ou dedicaram amizade a seu autor.

Esta tese é dedicada a três mulheres : Carolina, Rosilene e Margarida.

INDICE

RESUMO

São estudados Sistemas Interativos, sendo seus componentes identificados e descritos. É identificada a necessidade de ter-se uma linguagem apropriada para a especificação de estruturas de diálogos. Tal linguagem, chamada LINGUAGEM DE ESPECIFICAÇÃO DE DIALOGOS (LED), é definida. São apontadas vantagens da utilização de facilidades gráficas para a realização da comunicação entre operadores e programas de aplicação. O PROCESSADOR DE DIALOGO, uma ferramenta para definição, teste de protótipos e geração de estruturas de diálogos otimizadas é definido. Para apoio ao Processador de Diálogo, no tratamento gráfico, é justificada a utilização de um Núcleo Gráfico Padrão e, em decorrência, a existência de um GERENCIADOR DE DIALOGO, para exercer a supervisão da utilização do Núcleo Gráfico. As principais funções do Gerenciador de Diálogo são analisadas. São também analisadas características de diálogos e de operadores humanos. Ao final, é apresentado um exemplo de utilização da Linguagem de Especificação de Diálogos proposta.

ABSTRACT

Interactive Systems are studied. Its components are identified and described. The necessity of an appropriate language for dialogue structures specification is shown. This language is called DIALOGUE SPECIFICATION LANGUAGE (LED). Graphics facilities to support the communication between operators and application programs are discussed. A DIALOGUE PROCESSOR, a tool for definition, prototyping and generation of optimized dialogue structures, is defined. The utilization of a Standards Graphics Package for Dialogue Processor support in graphics treatment is justified; as a consequence, the necessity of a DIALOGUE MANAGER is shown. Dialogue Manager main functions are analysed. Dialogue and human operator characteristics are also analysed. At last, an utilization example of the proposed Dialogue Specification Language is presented.

1-INTRODUÇÃO

À época do aparecimento dos computadores e nos anos imediatamente posteriores, sua utilização era restrita aos especialistas em computação. Embora se atingisse cada vez maior progresso tecnológico, pouca atenção era dedicada à adequação dos computadores aos usuários. Assim, os usuários é que tinham de estar constantemente se adaptando aos aprimoramentos da tecnologia.

Com o passar do tempo, o custo dos sistemas computacionais caiu muito e seu uso extendeu-se por todas as áreas do conhecimento humano. A comunidade de usuários (não só no trabalho como também na vida particular) começou a se diversificar muito, o que passou a exigir a atenção dos especialistas para os problemas que surgiam na interface entre estes usuários não especialistas e o computador. Daí surgiram motivações para o estudo da interação homem-máquina, na expectativa de se encontrar uma forma de utilizar a potência da tecnologia existente para adaptar o computador ao homem.

Hoje em dia, a interação usuário-computador é de importância vital, já que existem mais e mais sistemas de informação sendo acessados pelas mais diversas pessoas. Se estes sistemas forem difíceis de entender e de operar eles causarão uma seleção de usuários, ou seja, terão seu uso restrito aos usuários especializados.

Assim, considera-se cada vez mais importante o estudo da interface existente entre o usuário de um sistema computacional e este sistema. Diversas conferências, encontros, oficinas de trabalho, enfocam o assunto, como por exemplo, os WORKSHOP's SEILLAC I (/GUED 79/), de Metodologias em Computação Gráfica, e SEILLAC II (/GUED 80/), de Metodologias em Interação, realizados em Seillac, na França, em 1976 e 1980, sessões nos congressos anuais da SIGGRAPH (comunidade norte americana de computação gráfica) e EUROGRAPHICS (similar européia) Workshop sobre Sistemas de Gerenciamento de Interfaces de Usuario (/SEEH 84/), em Seeheim, na Alemanha Federal, em 1984, Workshop's em técnicas de interação gráfica (patrocinados pela ACM/SIGGRAPH), realizados em Seattle, EUA, em 1981 (/GIIT 81/) e 1982 (/GIIT 83/), além de reuniões de grupos de trabalho das entidades de padronização nacionais (ANSI, dos EUA, DIN, da Alemanha Federal, etc) e internacional (ISO).

Todas estas reuniões têm contado com os mais diversos especialistas representantes dos mais diversos segmentos da Computação Gráfica : especialistas em computação, engenheiros e analistas oriundos de indústrias, "software e hardware houses", universidades, instituições de pesquisa; estudiosos de ciências humanas (psicólogos industriais e cognitivos); artistas; usuários. O objetivo destas reuniões tem sido a troca de informações e experiências dentro do campo e a tentativa de se

obter práticas comuns aceitas por todos os segmentos envolvidos.

Os primeiros resultados concretos em termos de consenso, foram a obtenção de um padrão internacional de software gráfico, o Núcleo Gráfico GKS, e as versões preliminares de padrões para transporte de dados de projeto (IGES) ou dados gráficos (CGM) e para interface entre pacotes gráficos independentes de dispositivos e os dispositivos (CGI).

Mais recentemente, importância maior tem sido dada à formalização de ferramentas para projeto de interfaces entre operadores e programas, que sejam "amigáveis" ao operador e, ao mesmo tempo, efetivas no sentido de garantirem o máximo desempenho do programa. A motivação para se atingir tal ferramenta veio da constatação, ao longo do tempo, de que, em geral, quando programadores de aplicação assumem sozinhos a tarefa de projetar o sistema interativo, sacrificam o entendimento do sistema pelos operadores em favor de uma suposta melhor performance do sistema, principalmente pelo fato de serem pessoas ligadas somente à aplicação, sem domínio das teorias de fatores humanos.

A partir desta constatação, observa-se uma tendência no sentido de considerar a tarefa de produção de sistemas interativos como afeita a um grupo interdisciplinar, dada a variedade de fatores envolvidos. Assim, seriam necessários os seguintes especialistas, conforme /GIIT 83/ :

- PROGRAMADOR DE APLICAÇÃO : responsável pelo desenvolvimento das partes específicas da aplicação;

- ESPECIALISTA DE APLICAÇÃO : estando intimamente familiarizado com a área de aplicação (é frequentemente um usuário em potencial) pode auxiliar o programador de aplicação;

- PROJETISTA DE INTERAÇÃO : responsável pelo projeto da interface de usuário; estando familiarizado com técnicas de interação e fatores humanos, pode discutir os requisitos de aplicação com o programador de aplicação. Para o projeto da interface, utiliza uma ferramenta de projeto de diálogos provida pelo gerenciador de ferramentas de diálogo;

- GERENCIADOR DE FERRAMENTAS DE DIALOGO : provê ferramentas de suporte do desenvolvimento de software de diálogo, isolando a aplicação dos vários mecanismos de interfaciamento com o usuário;

- USUARIO : provê realimentação aos outros especialistas, através de um processo iterativo, onde esta realimentação é utilizada para aprimorar a interface usuário/cistema interativo e o sistema interativo como um todo ("PROTOTYPING").

Quanto ao desempenho dos sistemas interativos, é necessário ter-se recursos para a realização de um monitoramento do sistema, no sentido de coleta e análise de dados sobre a interação, de forma a permitir avaliar o desempenho tanto do operador quanto do programa. Após a avaliação, o programa pode ser adaptado ao operador.

O presente trabalho está baseado em extensa pesquisa bibliográfica, em particular, da área de diálogos homem-computador, e na experiência adquirida com : (/SILV 85b/, /SILV 85c/)

- implantação de um núcleo gráfico segundo a norma GKS versão 6.2 nos computadores do DEE-FEC-UNICAMP;

- concepção de interfaces de entrada/saída entre o referido núcleo gráfico e terminais de vídeo (inclusive um terminal gráfico GT40);

- confecção de programas de aplicação para teste da operação do núcleo gráfico e para avaliação de técnicas de comunicação homem-máquina;

- participação na implementação de um núcleo gráfico segundo a norma GKS versão 7.2.

Seu objetivo é investigar os principais fatores e atividades envolvidos no projeto de software para sistemas interativos e, em particular, da interface entre o usuário e este sistema. Em decorrência desta investigação, são discutidas as principais características de uma ferramenta de software para auxílio no projeto, análise e implementação ("prototyping") de diálogos homem-computador, o PROCESSADOR DE DIÁLOGO. Tal ferramenta deve ser independente da aplicação específica, da linguagem de programação utilizada pela aplicação e dos dispositivos a serem acionados para entrada e saída de dados. São características suas, a possibilidade de definir e alterar a definição de um diálogo, supervisionar a execução, coletar dados sobre o desempenho, analisar estes dados e promover uma adaptação do diálogo ao operador específico. Em virtude da maior capacidade de apresentação de informações através de meios gráficos, e da consideração de que dados alfanuméricos representam um caso particular de dados gráficos, as entradas e saídas são tratadas como entradas e saídas gráficas e um núcleo gráfico dá suporte ao projeto da interface entre usuário e sistema. Para a representação do diálogo, o Processador tem como recurso a Linguagem de Especificação de Diálogos (LED) que permite a definição da estrutura ou das estruturas de qualquer diálogo. Esta linguagem é baseada no método de representação de estruturas de diálogo através de Células de Diálogo.

Na sequência deste capítulo, são enumerados os tópicos a serem apresentados em cada um dos capítulos subsequentes, sendo descrito o objetivo de cada capítulo.

O capítulo 2 enfoca a importância que deve ser dada à estrutura e controle de execução dos diálogos e as vantagens decorrentes da utilização de uma camada de diálogo independente de aplicação e dispositivos. São então discutidos os Sistemas Interativos de Computação. É apresentado um modelo de camadas para tais sistemas e cada camada é discutida em detalhe. São citados os aspectos relevantes da Interface de Usuário e do Diálogo, e os métodos comumentemente utilizados para a representação da estrutura de diálogos. O método de CELULAS DE DIALOGO é descrito em detalhe, em razão da opção em utilizá-lo como método de descrição de diálogos.

O objetivo do capítulo 3 é formalizar o Processador de Diálogo e tecer considerações para um futuro trabalho de implementação deste Processador. É então discutida a necessidade da existência de mecanismos para monitoramento, análise e adaptação de diálogos em uma ferramenta para tratamento completo de diálogos. É apresentada uma realização possível de tal ferramenta, denominada Processador de Diálogo. O Processador de Diálogo é definido sob o enfoque funcional e por sua topologia. A Linguagem de Especificação de Diálogos é definida. São apontadas e discutidas as principais consequências da utilização dos recursos gráficos de um Núcleo Gráfico Padrão por Sistemas para Tratamento de Diálogos.

No capítulo 4 são apresentadas as conclusões do trabalho e no capítulo 5 estão as referências bibliográficas.

Ao final, um apêndice apresenta um exemplo com o projeto do diálogo de um sistema gráfico interativo para desenho de figuras. São apresentadas as Células de Diálogo representando a estrutura proposta para o diálogo e as rotinas correspondentes em FORTRAN-77, como se fossem geradas pelo pré-processador. O objetivo é fixar e exemplificar os principais pontos apresentados nos capítulos 2 e 3.

2-SISTEMAS INTERATIVOS DE COMPUTAÇÃO

Sistemas computacionais interativos (com a interação sendo realizada através de entradas e saídas gráficas, alfanuméricas, sonoras, etc), caracterizam-se pela constante troca de informações entre usuário (operador) e sistema. O usuário fornece dados; como consequência, o sistema executa funções e apresenta respostas; o usuário fornece novos dados e este processo iterativo repete-se até o final da execução do programa. Ao contrário dos programas executados em "batch", para os quais o usuário prepara e fornece todos os dados anteriormente ao início da execução e não mais interfere até que esta execução tenha seu fim, nos sistemas interativos é dada a possiblidade ao usuário de contribuir dinamicamente para o desenrolar da execução do programa, podendo alterar a estratégia de execução ou desistir de opções que se mostrem ineficazes, otimizando a execução.

Um modelo de camadas de um sistema interativo pode ser representado por três componentes e duas interfaces, conforme a figura 2.1.

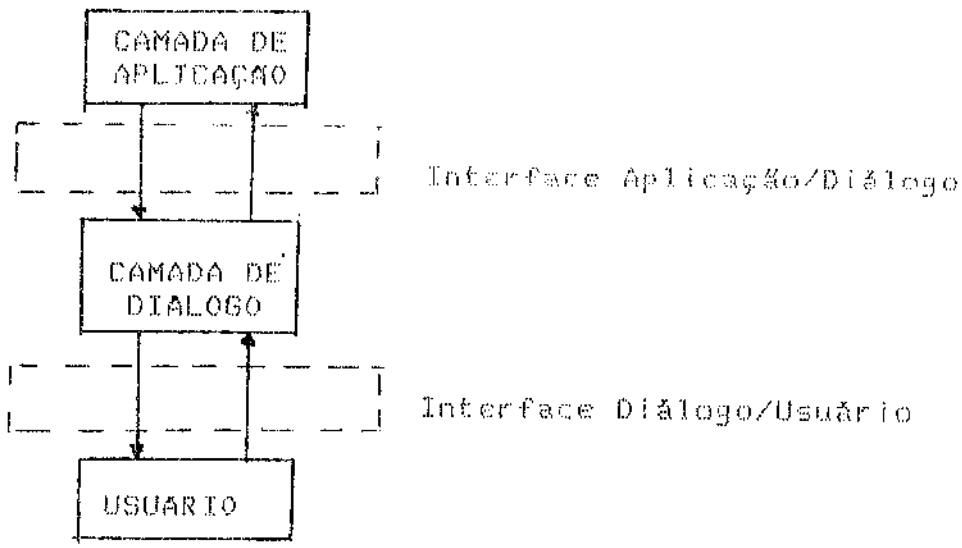


fig. 2.1 : Modelo de Camadas para Sistemas Interativos

A Camada de APLICAÇÃO é responsável pela solução dos problemas específicos da aplicação e pelo gerenciamento de dados da aplicação. Todas as informações necessárias a esta solução e a este gerenciamento são providas pelo USUARIO através da Camada de DIALOGO.

A Camada de DIALOGO é responsável pelo controle de toda a troca de informações entre usuário e sistema. Ou é acionada pelo usuário para fornecer dados à aplicação ou é acionada pela aplicação para proporcionar realimentação ao usuário.

A grande vantagem da existência de uma Camada explícita

de DIALOGO é tornar possível a separação entre a atividade de diálogo e a atividade a ser controlada pelo diálogo. Sendo a comunicação homem-máquina tratada isoladamente da aplicação, é possível utilizar um especialista em cada área para tratar da atividade correspondente. Este é um exemplo da coordenação interdisciplinar citada no capítulo 1.

O USUARIO é responsável pela operação do sistema interativo e dele se originam comandos e para ele são enviados resultados. É necessário que ele seja provido do conhecimento suficiente à operação deste sistema para poder executar com perfeição o seu trabalho.

A Interface Aplicação/Diálogo permite à Camada de DIALOGO o acesso às funções de aplicação e à aplicação o acesso às funções de diálogo responsáveis pela troca de dados com o usuário. Assim, esta interface define um mapeamento entre as primitivas do diálogo e as primitivas da aplicação.

A Interface Diálogo/Usuário representa a visão que o usuário tem do sistema e seu papel é importante porque ela é responsável pela apresentação ao usuário de um sistema fácil e agradável.

Quanto maior o grau desejado de independência entre a aplicação, os dispositivos e a forma de comunicação, maior o escopo e a complexidade das interfaces.

Daqui por diante não serão tratados casos especiais de informação como dados via voz. Considerando que dados alfanuméricos representam um caso particular de dados gráficos, serão tratados sistemas interativos via terminais gráficos, ou seja, sistemas onde as informações trocadas com o usuário são geradas por ou obtidas de dispositivos gráficos. Esta situação é cada vez mais frequente hoje em dia pois a utilização de informação gráfica na comunicação homem-máquina apresenta uma efetividade maior, já que dados que seriam impressos em forma numérica podem, com o auxílio da computação gráfica, ser plotados como gráficos, dispostos em mapas, desenhados como diagramas ou exibidos como cenas tridimensionais. Além disso, o potencial dos dispositivos gráficos não está somente na velocidade com a qual eles podem gerar tais imagens mas também na flexibilidade, ou seja, na habilidade de representar a mesma informação em uma variedade de formas diferentes. Deve-se notar que o oferecimento de facilidades gráficas para a realização da comunicação de um usuário com um sistema interativo representará um "overhead" ao sistema se somente se pretendem realizar diálogos alfanuméricos.

Isto posto, surge uma importante tarefa para a Camada de DIALOGO que é a geração de figuras e texto (como formas de apresentação de opções de escolha e de resultados gráficos ao usuário) e a interpretação de dados gráficos gerados pelo usuário (comandos e dados). Novamente, a complexidade do sistema como um todo e os graus de independência e portatilidade pretendidos são decisivos na definição da Camada de DIALOGO. Para sistemas

simples, esta própria camada poderia ser o gerador e coletor de dados gráficos e para tal sua implementação incluiria todas as preocupações com os dispositivos específicos a serem utilizados.

Por outro lado, seria interessante que a Camada de Diálogo realizasse somente a supervisão desta tarefa de geração e interpretação de dados gráficos, utilizando então os recursos de um Núcleo Gráfico para a sua execução, o que tornaria esta camada conceitualmente independente dos dispositivos.

Com esta visão, extende-se o modelo de camadas para o sistema interativo, e o resultado é apresentado na figura 2.2, incluindo um Núcleo Gráfico e os dispositivos.

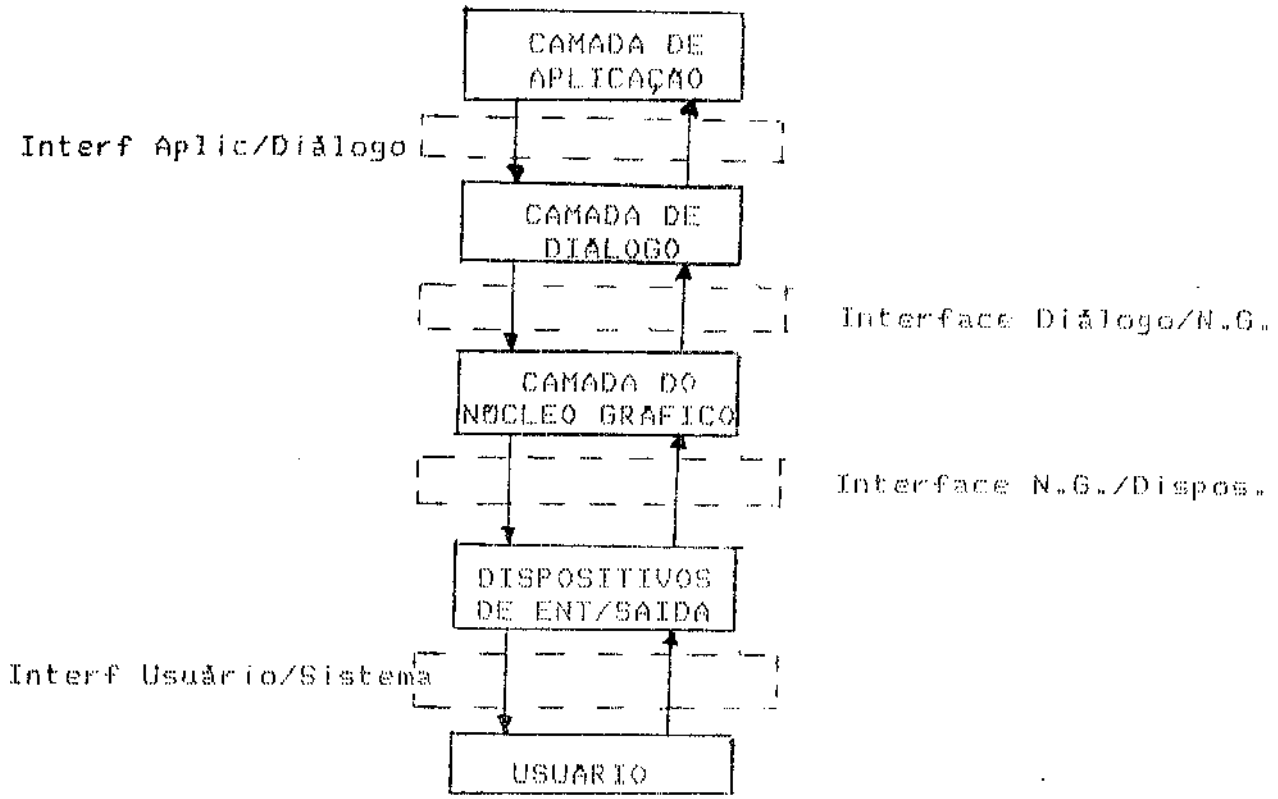


fig. 2.2 : Modelo extendido de camadas para Sistemas Interativos

A formalização de todas as interfaces apresentadas na figura 2.2 é importante justamente para permitir uma modularização do sistema com tratamento individualizado de cada bloco, de acordo com a atividade por ele realizada. A complexidade de cada uma das interfaces é tanto maior quanto maior for o grau de independência pretendido para um bloco em relação a outro.

A seguir, cada componente e cada interface apresentados na figura 2.2 é descrito detalhadamente.

2.1-CAMADA DE APLICAÇÃO

A gama de aplicações que se utiliza de computadores para efetivar suas necessidades é enorme. Podem ser citadas as áreas de Controle de Processos, Projetos de Peças Mecânicas, Produção de Desenhos, Automação de Escritórios, Apoio à Pesquisa, como exemplos destas aplicações.

A camada de aplicação engloba as rotinas que se incumbem da função primordial do sistema, ou seja, a realização das tarefas desejadas pelo usuário. A aplicação se utiliza dos recursos do diálogo para obter dados do usuário e enviar respostas a ele mas é a própria aplicação que faz uso dos dados recebidos e decide as respostas correspondentes.

Esta divisão, entre o tratamento da comunicação e a utilização da informação, permite que especialistas da área de aplicação construam sua camada independentemente da construção da camada de diálogo. Desta forma, a camada de aplicação pode ser projetada e inclusive testada separadamente e a utilização de uma camada pela outra se fará através da definição de uma interface que atenda aos requisitos das duas partes. Esta interface deve ser definida em conjunto pelos dois especialistas, após o que, a camada de aplicação pode ser construída supondo as interações a serem realizadas com o usuário tratadas por rotinas já existentes ou implementáveis independentemente.

2.2-CAMADA DE DIALOGO

A camada de diálogo é responsável pelo controle do fluxo de informação que se origina na aplicação e se destina ao operador, ou vice-versa, isto é, é responsável pelo controle do diálogo.

Um diálogo envolve o compartilhamento de conhecimento através da troca de informação. Diálogos humanos variam enormemente em sua duração e na complexidade e riqueza da informação trocada. Diálogos eficientes requerem um meio de comunicação transparente, ou seja, o meio não deve apresentar restrições ao diálogo, seja pela natureza, tamanho ou frequência da comunicação entre os participantes. Como esta é uma situação ideal, pois na prática sempre haverá restrições, deve-se procurar minimizar estas restrições. A eficiência do diálogo também depende da compreensão e consequente reação de um participante à realimentação recebida do outro. Esta realimentação deve ser utilizada para modificar a comunicação, de forma que ela melhor se adapte ao participante que gerou a realimentação, principalmente quando este participante apresenta dificuldades em entender o outro.

Estas características de diálogos humanos também estão presentes em diálogos homem-computador. Entretanto, neste segundo caso, o meio de comunicação e a capacidade de resposta de pelo menos um dos participantes são limitados. Apesar disto, um bom e consciente projetista de diálogos pode obter um grau considerável de riqueza e eficiência em seus sistemas. Deve ser previsto que algumas pessoas serão mais experientes que outras quando lidando com o computador. A estrutura do diálogo não deve então ser excessivamente complexa, a ponto de tornar o programa difícil de ser operado pelos menos experientes, e nem excessivamente detalhada, a ponto de tornar a operação do programa uma tediosa tarefa aos mais experientes.

É fato que não existe uma metodologia ideal que se aplique ao projeto de qualquer tipo de diálogo. Assim, a literatura científica limita-se a recomendar diretrizes a serem seguidas na tentativa de se obter um diálogo eficiente.

São encontradas na literatura, contribuições de especialistas que estudaram e obtiveram experiência na tarefa de construção de interfaces de usuário, ou seja, de diálogos homem-computador (/DEHN 81/, /STEW 80/, /SHAC 80/, /NEWM 81/, por exemplo). Os pontos citados por estes especialistas, abordados no item 2.5.4, devem ser considerados em tais projetos, sempre tendo em mente um perfeito compromisso entre a eficiência do produto final e o impacto que o diálogo causa no usuário, ou seja, quão fácil e agradável é para o usuário a operação do sistema. O ato de definir um diálogo representa o ato de fixar a visão, ou o entendimento, que o usuário tem do sistema como um todo, pois é

através do diálogo que o usuário toma conhecimento do sistema.

Uma vez definido o diálogo, consideradas todas as técnicas e preocupações referidas acima, surge a necessidade de se encontrar uma forma para a representação deste diálogo, ou seja, uma forma de abstrair a estrutura do diálogo sendo projetado, para que este diálogo possa ser formalizado.

Para poder controlar então o diálogo entre operador e sistema, a camada de diálogo necessita de uma descrição de todas as ações que o usuário pode escolher e das respostas que a aplicação pode oferecer. Isto não é mais do que a própria estrutura do diálogo que deve representar, para cada estado possível da aplicação, as entradas permitidas, as funções a ativar (transição para outros estados) e as respostas possíveis.

Diversos métodos podem ser utilizados para a representação da estrutura de diálogos. A escolha de um deles é influenciada por três fatores (/HANU 80/):

- capacidade de representar desde as mais simples às mais complexas estruturas;

- capacidade de descrição de diferentes sequências de diálogo para uma mesma função de aplicação;

- facilidades de que o método dispõe para a implementação das rotinas do diálogo e ativação das funções de entrada/saída.

As formas de representação da estrutura de diálogos auxiliam na tarefa de definir e formalizar os passos do diálogo.

Um estudo destes métodos sugere sua classificação em três grupos, de acordo com a potencialidade de representação.

2.2.1-GRUPO 1 : Representação Através de Transições Entre Estados Simples

O grupo 1 é caracterizado pela representação da estrutura dos diálogos através de estados simples e transições entre estes estados. A interação entre sistema e usuário é vista como estando a cada momento em um estado, caracterizado por um conjunto de mensagens de entrada (ou condições) possíveis e por um conjunto de estados internos, decorrentes das transições ocorridas anteriormente, e suas interpretações geram transições para outros estados. Poderá estar vinculada a cada mensagem de entrada uma mensagem de saída que sirva de realimentação ao operador ou que seja o próprio resultado de algum processamento. Pertencem a este grupo : ARVORES DE ESCOLHA, DIAGRAMAS DE TRANSIÇÃO DE MAQUINAS DE ESTADOS FINITOS e REDES DE PETRI.

2.2.1.1-Árvores de Escolha (/MORE 82/, /DLAC 77/)

Esta forma de representação de estruturas considera o diálogo como uma árvore, com nós correspondendo aos pontos nos quais o usuário tem uma escolha e ramos selecionados a partir das mensagens de entrada, e que controlam as mensagens de saída e ações a serem executadas. A árvore completa representa a estrutura do diálogo e caminhos através dela são sequências de entrada permitidas ao usuário.

2.2.1.2-Diagramas de Transição de Máquinas de Estados Finitos (/PARN 69/)

Considerando que a interação entre usuário e computador está sempre em um estado específico, associado a um conjunto de mensagens de entrada e de saída possíveis, tem-se uma similaridade com a teoria das máquinas de estados finitos e pode-se usar um diagrama similar para representar a estrutura da interação.

A figura 2.3 apresenta o primeiro estado de um diálogo (o estado inicial), quando o usuário pode dar um comando de entrada no sistema ("LOGON"), ao qual o sistema responde com a mensagem correspondente, por exemplo, a classe do usuário (#1 ou #2) e efetua a transição correspondente; ou o usuário pode simplesmente solicitar informações sobre o carregamento ("LOAD") ou avisos ("NEWS?"), quando então não há transição de estado.

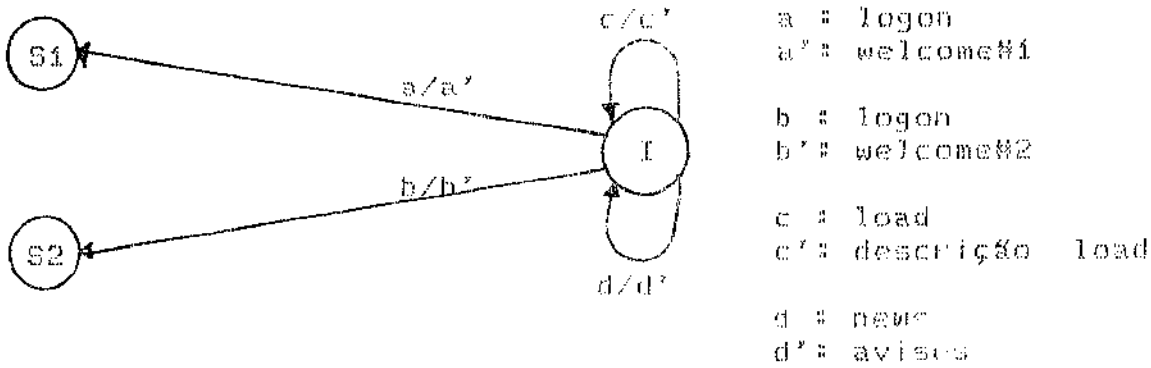


fig. 2.3 : Diagrama de transição para o exemplo.

2.2.1.3-Redes de Petri (/PETE 77/)

A Rede de Petri é um grafo dirigido que possui dois

tipos de elementos :

- EVENTOS ou TRANSIÇÕES : representados por barras verticais;

- CONDIÇÕES ou LOCAIS : representados por círculos.

Um evento é dito habilitado se todas as suas condições de entrada estão satisfeitas. Se o evento acontece (ou seja, ocorre a transição), as condições de saída verificam-se. Com esta regra, podem ser analisadas todas as seqüências de eventos de um grafo, supondo-se as entradas de um evento satisfeitas, a ocorrência da transição, a validade das condições de saída que então podem ser consideradas como condições de entrada para os outros eventos, e assim por diante.

A figura 2.4 mostra com simplicidade de detalhes um aspecto da interação entre um usuário e um interpretador de comandos. Supondo inicialmente o interpretador em estado de espera, ao usuário fornecer um comando, estar-se-á habilitando a transição T1 que, ao ocorrer, torna verdadeira a condição de saída (ou seja, o comando é interpretado). Esta passa a ser a condição de entrada das transições T2 e T3 e, estando verificada, ambas as transições estão habilitadas. Supondo-se ora a transição T2 ocorrendo para um comando válido (com consequentes execução da ação, devolução de resposta ao usuário e retorno ao estado de espera), ora a transição T3 ocorrendo para um comando inválido (com consequentes emissão de mensagem de erro ao usuário e retorno ao estado de espera), são analisadas as possíveis seqüências de eventos.

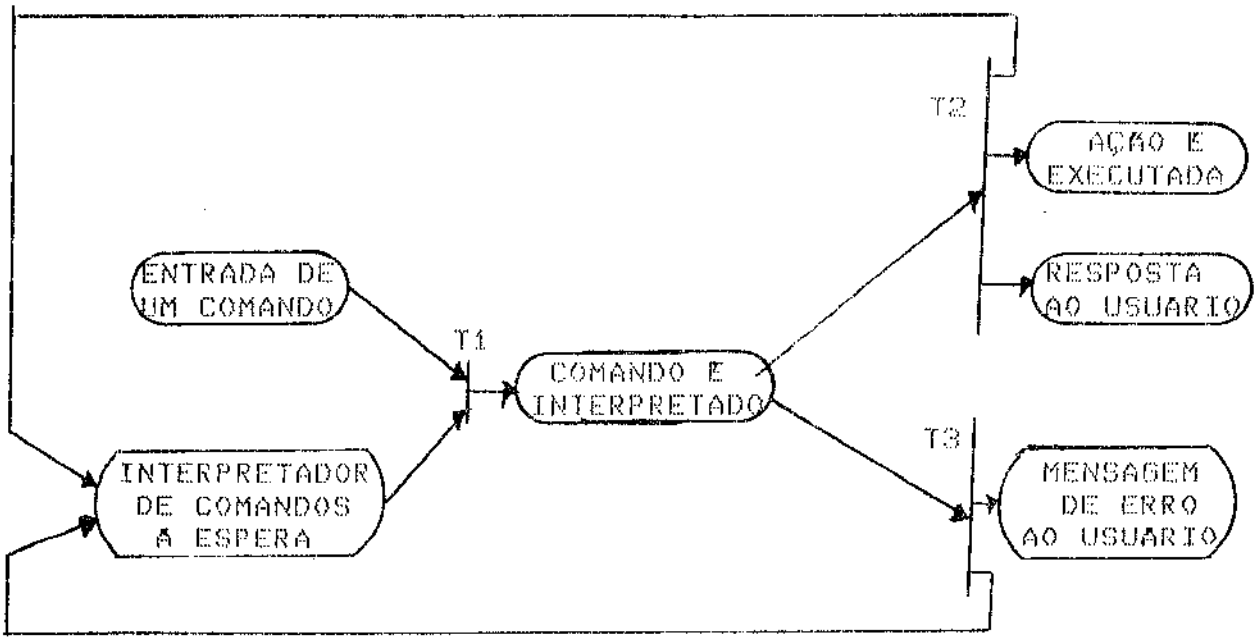


fig. 2.4 : Rede de Petri para o exemplo dado.

2.2.2-GRUPO 2 : Representação Através de Transições Entre Estados Complexos

No grupo 2 aparecem métodos cuja principal característica é a possibilidade de descrição de subdiálogos, ou seja, estados complexos de um diálogo principal são descritos em detalhe através de outros diagramas. Com isto, estes métodos se prestam à tão conhecida metodologia de projeto "top-down", ou seja, de refinamento passo a passo. Assim, estruturas complexas são descritas através de estruturas cada vez mais simples. Os métodos em destaque neste grupo são DIAGRAMAS MODIFICADOS DE TRANSIÇÃO ENTRE ESTADOS e GRAMÁTICAS FORMAIS.

2.2.2.1-Diagramas Modificados de Transições Entre Estados (/DENE 77/, /ENCA 82/, /HANU /)

São métodos que aplicam variações do diagrama de transição de estado das máquinas de estados finitos, permitindo projeto "top-down", através de estruturação hierárquica dos diagramas de estado. São criados símbolos especiais para representar os estados inicial e final, estado simples, estado complexo (um subdiálogo, ou seja, um estado que é representado por outro diagrama detalhado), ponto de interação (onde o usuário fornece uma entrada), comentários e a própria transição.

Denert (/DENE 77/) definiu os símbolos apresentados na figura 2.5, para uso nos diagramas modificados de transição.

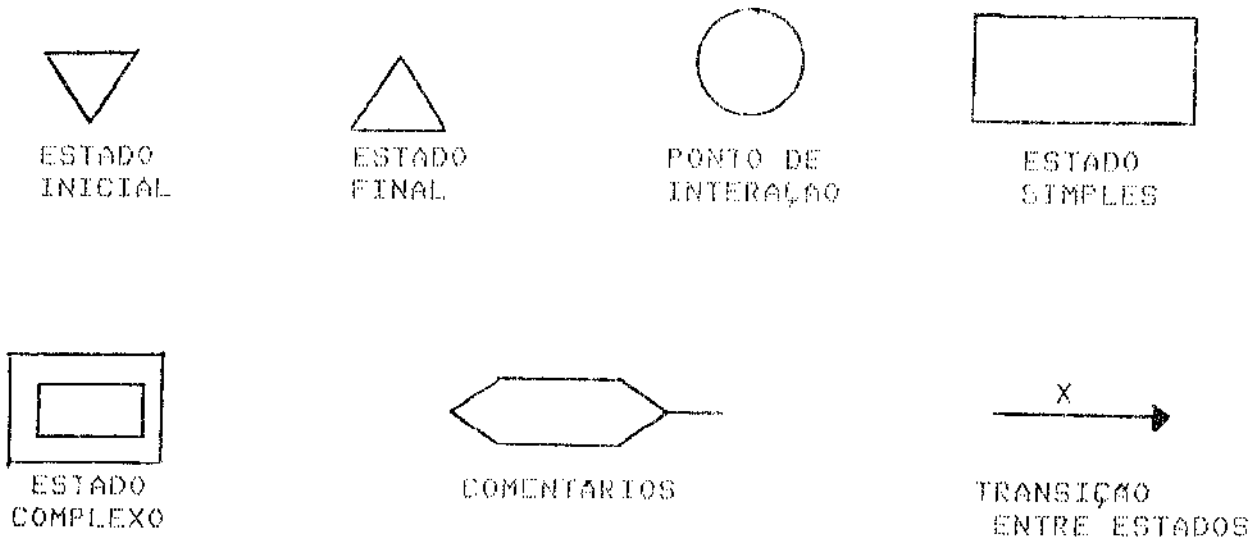
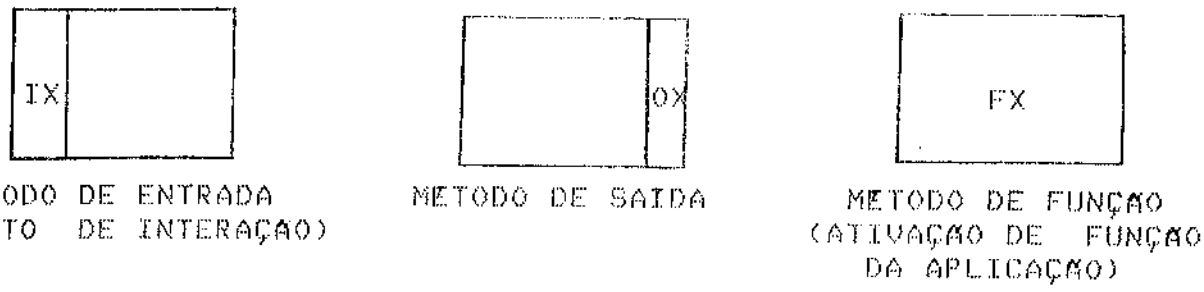
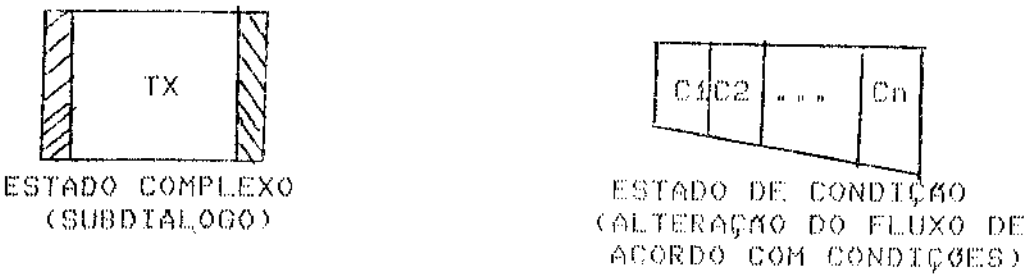


fig. 2.5 : Conjunto de símbolos para diagramas de transição modificados.

Em /ENCA 82/ foi definido outro conjunto de símbolos, apresentado na figura 2.6, onde o X presente em cada símbolo é um número que identifica um símbolo específico dentre um conjunto de símbolos do mesmo tipo.





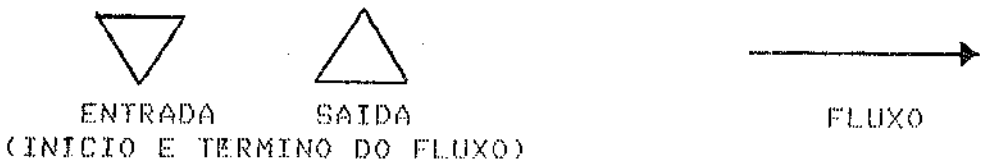


fig. 2.6 : Conjunto alternativo de símbolos para diagramas modificados.

A figura 2.7 mostra a representação de um passo na interação entre um programa e um usuário, através de um diagrama modificado de transição entre estados, utilizando os símbolos definidos na figura 2.6.

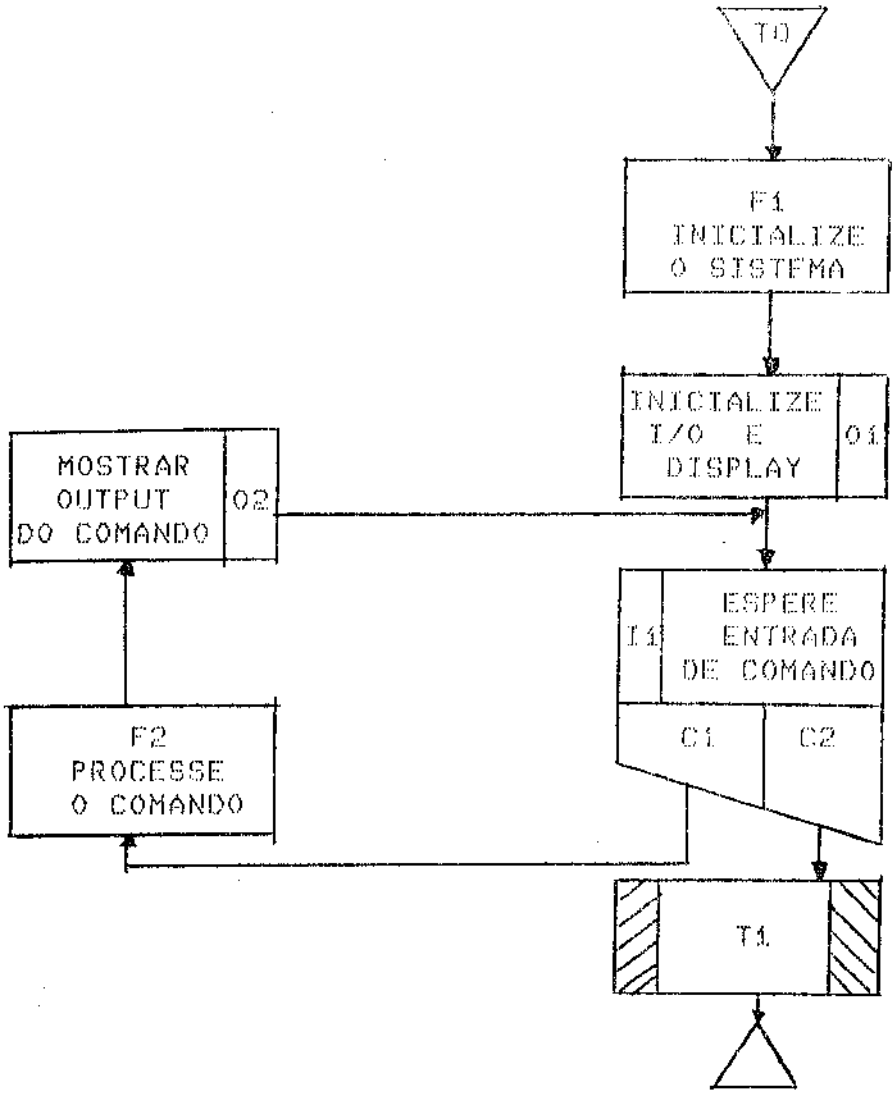


fig. 2.7 : Representação de um passo na interação programa-usuário.

2.2.2.2-Gramáticas Formais (/REIS 79/)

Da mesma forma que para as linguagens utilizadas na comunicação entre as pessoas, para as linguagens de programação, gramáticas são necessárias para descrever as regras a serem obedecidas na formação de sentenças. Em ambos os casos, é utilizada uma descrição formal da sintaxe da linguagem, o que permite a construção de sentenças gramaticalmente corretas.

No caso dos sistemas interativos, o que se pretende é a descrição do diálogo, ou seja, das ações que o usuário realiza ao interagir com um programa. Para isto, utiliza-se um conjunto de regras de produção que permitirão a construção de cadeias de caracteres através de diagramas em árvore baseados nestas regras.

Uma gramática consiste de :

- um conjunto de símbolos terminais (as palavras na linguagem);

- um conjunto de símbolos não terminais (ou seja, símbolos que podem ser descritos através de um ou mais dos símbolos terminais e de outros símbolos nao terminais);

- um símbolo de início (ex.: S para sentença);

- metasímbolos (símbolos especiais para definição dos símbolos não terminais ou para concatenação de símbolos terminais e não terminais);

- regras definidas com os componentes acima (regras de produção).

No caso dos diálogos, os símbolos terminais representam as próprias ações que o usuário executará. Os símbolos não terminais são construções criadas para mostrar a estrutura do diálogo; eles representam conjuntos de ações similares que podem ser agrupadas e descritas de uma mesma forma. Exemplos de metasímbolos são :

- "::=" significando "é composto de";

- "/" significando "ou"(separador de opções);

- "+" significando "e"(concatenação).

Exemplos de regras :

- <picture> ::= <colored shape> / <picture> + <colored shape>

- <colored shape> ::= <color> + <shape> / <shape> + <color>

- <color> ::= <new color> / <old color> / <default color>

- <new color> ::= CURSOR IN RED / CURSOR IN BLUE / CURSOR IN GREEN

Os símbolos entre colchetes(< e >) são os não terminais e os em letras maiúsculas os terminais(CURSOR IN RED significaria então uma ação do usuário que é o apontamento, através do cursor, da cor RED).

2.2.3-GRUPO 3 : Representação por Células de Diálogo (/BORU 81/, /BORU 82/, /HANU 82/, /ENCA 83/)

Embora a representação por Células de Diálogo possa ser

considerada sob o ponto de vista de Transições entre Estados Complexos (principalmente em relação a Gramáticas Formais), este método acrescenta, em relação aos métodos do grupo 2, recursos específicos para diálogo. Assim, optou-se por classificá-lo separadamente do grupo 2.

O diálogo entre o usuário e o computador é descrito através de quatro elementos básicos :

- PROMPT : elemento que efetua a solicitação ao usuário para fornecer algum dado (normalmente apresentado como uma indicação do tipo de dado requerido);

- SYMBOL : elemento que representa a própria entrada do usuário;

- ECHO : elemento que apresenta ao usuário a interpretação do símbolo por ele fornecido;

- VALUE : elemento resultante da interpretação do símbolo, sendo liberado pelo diálogo para uso posterior pela aplicação.

A Célula de Diálogo é definida como uma unidade que descreve totalmente um passo de um diálogo, o qual consiste de todas as ações necessárias à transferência de um elemento de diálogo de um participante ao outro. Estas ações envolvem o fornecimento ao usuário de toda a informação necessária à execução de sua tarefa, a avaliação das entradas do usuário, a emissão de realimentação, a obtenção do valor a partir das entradas e a liberação do valor final. Assim, a célula reunirá em si os quatro elementos: PROMPT, SYMBOL, ECHO e VALUE. Para entrada utilizam-se células pré-definidas que, na realidade, representam funções de entrada de um sistema de base utilizado como suporte ao sistema de diálogo.

Subcélulas de Diálogo podem ser ativadas para a realização de ações específicas, o que permite então o refinamento passo a passo do diálogo.

Com base na teoria de Células de Diálogo, exposta em /BORU 81/, e na análise de exemplos de utilização desta forma de representação de estrutura de diálogos, apresenta-se em seguida uma interpretação deste método, adaptada às necessidades do Processador de Diálogo a ser exposto no capítulo 3.

Cada Célula de Diálogo é formada por CABEÇA e CORPO.

Fazem parte da CABEÇA :

- o NOME da célula, ou seja, sua identificação;

- a LISTA de parâmetros;

- um bloco de DECLARAÇÃO DOS PARAMETROS (dados

classificáveis temporalmente em relação ao fluxo de dados entre as células), que podem ser :

- de entrada (IN) : parâmetros gerados em uma célula anterior e passados à célula em questão quando de sua ativação;

- de saída (OUT) : parâmetros gerados na célula em questão e retornados à célula que a ativou;

- transientes (TRANS) : parâmetros que vieram da célula anterior ou foram definidos na célula atual e, modificados ou nao, retornarão a esta célula ou serão enviados à célula posterior. Na verdade, representam parâmetros de entrada/saída.

- um bloco de DECLARAÇÃO DOS DADOS (dados classificáveis em relação ao ambiente em que são manipulados ou definidos), que podem ser :

- globais (GLOB) : dados cujo ambiente transcende uma determinada célula. Podem ter seus valores lidos ou escritos em qualquer célula na qual estejam declarados;

- locais (LOC) : dados definidos e tendo atribuição de valores na célula em questão. Se forem transmitidos a outras células, transformam-se em parâmetros de entrada das células, nelas só podendo ser lidos.

- um bloco de INICIALIZAÇÃO (INIT), onde valores iniciais, se necessário, são fornecidos a variáveis e parâmetros do tipo INTEGER, REAL, BOOLEAN, STRING, POINT e são definidos dados dos tipos PICTURE, PROMPT, ECHO, VIEWPORT para representar :

- áreas na tela de display para comunicação com o usuário através de prompts, ecos, menus, etc. (<viewp def>);

- figuras para representar graficamente prompts e ecos (<pict def>);

- prompts e ecos (<prompt def> e <echo def>).

O principal objetivo das classificações de parâmetros e de dados é possibilitar a execução de testes semânticos, durante a pré-compilação de cada célula, já que, da própria formalização surgem as restrições quanto à utilização de cada dado ou parâmetro dentro da célula, e inconsistências e erros podem ser verificados (ex.: parâmetros de entrada não podem aparecer do lado esquerdo de expressões, enquanto os de saída nao podem aparecer do lado direito).

A hierarquia de células, quanto à passagem de parâmetros, está exposta na figura 2.8.

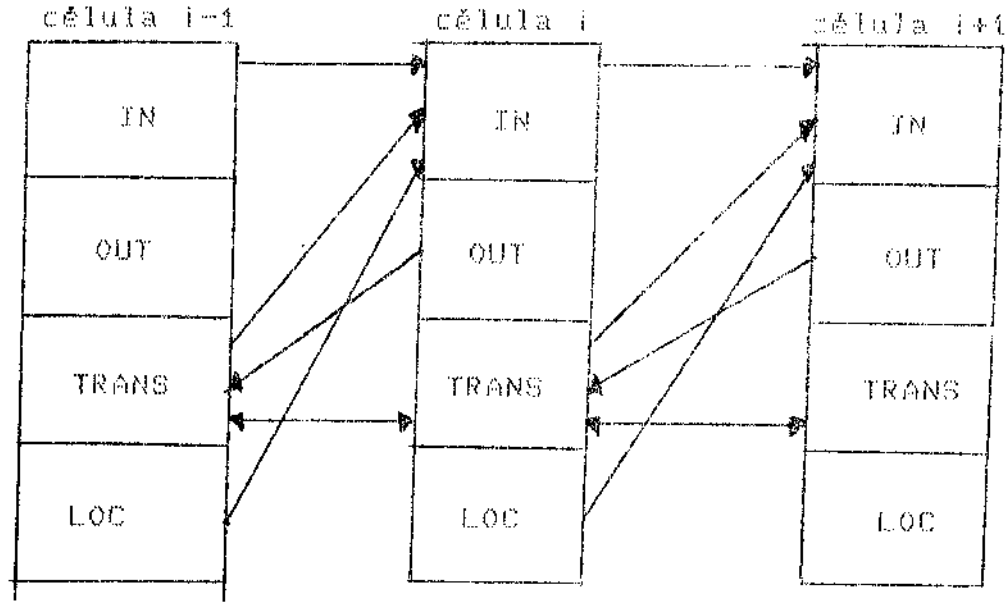


fig. 2.8 : Hierarquia entre Células de Diálogo.

No CORPO da célula (BODY), está a definição da execução do diálogo, ou seja, a especificação da função da célula, que pode ser :

- modificação de atributos, definição ou deleção de prompts (<prompt mod>, <prompt def> ou <prompt del>);

- modificação de atributos, definição ou deleção de ecos (<echo mod>, <echo def> ou <echo del>);

- ativação de subcélulas para entrada de dados ou para atividades específicas (SYMBOL < name>);

- computações sem envolver interação (<calculation>).

O método de Células de Diálogo utiliza textos e/ou figuras para a comunicação gráfica com o operador. Os elementos de comunicação permitidos nas Células de Diálogo são PROMPT's e ECHO's. Assim, cada PICTURE ou TEXT deve estar sempre contida em uma PROMPT ou ECHO, e assumir então os atributos do respectivo elemento.

Figuras e texto são gerados utilizando um sistema de coordenadas normalizadas (0,0 a 1,1). Para sua exibição, as coordenadas são transformadas de forma que os pontos (1,0) e (0,1) sejam mapeados aos vértices da diagonal principal da viewport associada, ou seja, as coordenadas originais são relativas à superfície de exibição como um todo mas as coordenadas de exibição relacionam-se, proporcionalmente, à sua janela de exibição. Se a viewport na qual figuras ou texto devem ser gerados não é declarada, é subentendida a viewport default, que representa a tela completa (0,0;1,1).

Para a representação das Células de Diálogo, será utilizada notação gramatical formal (Metalinguagem), com os seguintes metasímbolos:

- < > : símbolo não terminal, ou seja, que pode ser decomposto em símbolos não terminais e/ou terminais;

- / : separa alternativas para a definição de um símbolo;

- ( ) : engloba uma expressão sobre a qual atua um operador;

- * : operador de repetição (0 ou mais vezes);

- + : operador de repetição (1 ou mais vezes);

- 0 : inexistência da expressão;

- ::= : definição de um metasímbolo por uma metaexpressão, indicando que o lado esquerdo da expressão pode ser substituído pelo lado direito em qualquer local que ele apareça;

- símbolos terminais são palavras chaves (reservadas), que não podem ser decompostos, e serão aqui representados em letras maiúsculas.

Uma Célula de Diálogo pode ser representada então, como a seguir, onde valores sublinhados numa definição de símbolo representam valores default, ou seja, valores que serão tomados caso não seja declarado um valor para o símbolo :

```
<dial cell> ::=

DIACELL < name> (<name>*) / 0

      IN    (<type>/<ppev type>  <name> (<dimension>)    ))*   / 0
      TRANS (<type> <name> (<dimension>) / 0)*                 / 0
      OUT   (<type> <name> (<dimension>) / 0)*                 / 0
      PDECL END

      LOC   (<type>/<ppev type>  <name> (<dimension>) / 0)*    / 0
      GLOB  (<type> <name> (<dimension>) / 0)*                 / 0
      DDECL END

      INIT <value init>
           (<view def> / <pict def> / <prompt def> / <echo def>)
      INIT END     /  0

      BODY (<prompt def> / <prompt mod> / <prompt  del>) /
           (SYMBOL <name> (<name>*)/0  ) /
           (<echo def> / <echo mod> / <echo del>) /
           (<calculation>)

END DIACELL
```

onde :

- <name> é o identificador de uma célula ou de um parâmetro sendo passado a uma célula. No caso dos elementos PICTURE, PROMPT, ECHO e VIEWPORT, o diálogo gera os identificadores e os fornece nas variáveis definidas;

- < type> ::= INTEGER / REAL / BOOLEAN / STRING / POINT

representa o tipo de dado e os tipos válidos são INTEGER, REAL, BOOLEAN, STRING, comuns nas linguagens de programação e o tipo POINT, apropriado para especificação das duas coordenadas de um ponto. O acesso a cada uma das coordenadas é possível mediante a expressão <name>.X para abscissa e <name>.Y para ordenada;

- <ppev type> ::= PICTURE / PROMPT / ECHO / VIEWPORT

representa novos tipos adicionados para atenderem às representações gráficas : PICTURE, PROMPT, ECHO, VIEWPORT. Variáveis definidas como PICTURE, PROMPT, ECHO e VIEWPORT representam apontadores para os objetos, ou seja, quando são definidas o sistema de diálogo se responsabiliza por atribuir a cada uma delas um valor que represente o objeto. Assim, se o autor de diálogo necessitar referenciar um determinado objeto, o valor da variável por ele definida será a referência. Em virtude disso, variáveis destes quatro tipos só podem ser acessadas para leitura e nunca para escrita, razão pela qual nunca podem ser definidas como parâmetros de saída ou transientes e nem como dados globais;

- <dimension> é um valor inteiro que representa a dimensão de arranjos;

- <value init> representa a atribuição de valores iniciais para parâmetros ou dados de qualquer dos tipos permitidos em <type>;

- <view def> ::= VIEWPORT DEF <name> <real>4 DEF END

- Uma viewport é definida por um identificador e quatro valores reais (coordenadas dos dois pontos inferior esquerdo e superior direito do quadrado, na ordem xmin, ymin, xmax, ymax);

- <pict def> ::= PICTURE DEF <name> VIEW<name>/0 <attribut>
<output> DEF END

- Para definir uma figura são necessários um identificador, a viewport onde ela será exibida, seus atributos (visibilidade e cintilamento) e as funções de saída responsáveis pela construção de sua representação gráfica (<output>);

- <attribut> ::= (VISIBLE / INVISIBLE) / (STEADY / BLINK)

- ⟨prompt def⟩ ::= PROMPT DEF ⟨name⟩ VIEW⟨name⟩/0 ⟨duration⟩ (⟨attribut⟩ ⟨prompt obj⟩) / (MENU FOR CHOICE ⟨integer⟩ ⟨menu attr⟩ ⟨menu obj⟩) DEF END

- Para definir prompts são necessários um identificador, uma viewport associada, atributos (visibilidade e cintilamento), duração (até a próxima prompt ou eco, até a próxima prompt ou eco na mesma viewport (SVPT) ou até a desativação da célula) e a especificação do objeto que servirá de prompt (uma picture já definida anteriormente, uma cadeia de caracteres em uma dada posição ou um menu de escolhas). Um menu significa uma opção de um valor entre um grupo de valores (range), já que ele está associado a um dispositivo de entrada do tipo CHOICE, e pode ser representado graficamente por figuras (pictures) ou strings (TEXT's);

- ⟨duration⟩ ::= TILL NEXT ECHO / TILL NEXT PROMPT / TILL NEXT ECHO SVPT / TILL NEXT PROMPT SVPT / TILL CELEND

- ⟨prompt obj⟩ ::= PICTURE ⟨name⟩ / TEXT ⟨string⟩

- ⟨string⟩ ::= POINT STRING

- ⟨menu attr⟩ ::= (VISIBLE / INVISIBLE) (⟨integer⟩:⟨integer⟩)/ ⟨integer⟩*/0

- ⟨menu obj⟩ ::= PICTURE ⟨name⟩ * / TEXT ⟨string⟩*

- ⟨echo def⟩ ::= ECHO DEF ⟨name⟩ VIEW⟨name⟩/0 ⟨duration⟩ ⟨attribut⟩ ⟨echo obj⟩ DEF END

- Para definir ecos : identificador, viewport associada, atributos (visibilidade, cintilamento), duração, especificação do objeto que é utilizado como eco (uma string de caracteres ou uma picture ou uma prompt já definidas anteriormente);

- ⟨echo obj⟩ ::= PICTURE ⟨name⟩ / PROMPT ⟨name⟩ / TEXT ⟨string⟩

- ⟨prompt mod⟩ ::= PROMPT MOD ⟨name⟩ (VISIBLE/INVISIBLE) / (STEADY / BLINK) / TRANSFOM ⟨real⟩ 6 MOD END / PROMPT MENU MOD ⟨name⟩ ⟨menu attr⟩ MOD END

- ⟨echo mod⟩ ::= ECHO MOD ⟨name⟩ (VISIBLE / INVISIBLE) / (STEADY / BLINK) / TRANSFORM ⟨real⟩ 6 MOD END

- As modificações de atributos de prompt e eco significam modificação de visibilidade ou modo de cintilamento ou transformações de suas coordenadas. As transformações são expressas através de uma matriz que concatena rotação, translação e escalamento e não são cumulativas, ou seja, a matriz unidade recupera as coordenadas originais;

- <prompt del> ::= PROMPT DELETION <name>

- <echo del> ::= ECHO DELETION <name>

- Para a deleção de prompts ou ecos só são necessários seus identificadores.

- <calculation> representa qualquer processamento que não envolva interação.

No apêndice um exemplo ilustra a utilização de Células de Diálogo.

2.3-DISPOSITIVOS DE ENTRADA/SAIDA E CAMADA DO NUCLEO GRAFICO

A COMPUTAÇAO GRAFICA, segundo Foley (/FOLE 82/), envolve a criação, o armazenamento e a manipulação de modelos de objetos e de suas figuras via computador. A COMPUTAÇAO GRAFICA INTERATIVA é o importante caso particular da Computação Gráfica, em que o usuário controla dinamicamente conteúdo, formato, tamanho ou cores de figuras na tela, por meio de dispositivos de interação como teclado, light-pen ou joystick.

Uma motivação para o desenvolvimento de técnicas e equipamentos relativos ao campo da computação gráfica tem sido o fato de ela ser um instrumento extremamente efetivo de comunicação entre o homem (usuário) e o computador (sistema de computação). Tanto a quantidade de informação que o usuário pode absorver de gráficos ou figuras, quanto a velocidade de absorção desta informação mostram-se superiores quando a saída gráfica é comparada a outras formas convencionais. Por outro lado, se figuras estáticas já são um bom meio de transmitir informações, figuras modificáveis dinamicamente são um meio melhor.

A literatura apresenta trabalhos desde 1963 que documentam grande aumento na qualidade da interação homem-máquina a partir do uso de técnicas gráficas (/SUTH 63/, /ROSS 63/). Mas, àquela época havia pouco desenvolvimento na área de hardware gráfico e os dispositivos eram caros.

Com o desenvolvimento da microeletrônica, os equipamentos utilizados no campo da Computação Gráfica não somente foram bastante aprimorados como também sofreram diminuição acentuada de custo, o que contribuiu para uma maior difusão de seu uso e, consequentemente, para a criação de novas técnicas de geração e manipulação de figuras.

Atualmente, tem-se diversos tipos de dispositivos gráficos que servem às mais diversas necessidades, desde a saída passiva através de plotters, até o trabalho interativo através de terminais que permitem tanto a exibição de dados num vídeo, quanto o fornecimento de informações de entrada para o sistema. Os dados exibidos são gerados por um processador que lê uma memória de instruções, o DISPLAY-FILE, interpreta-a e desloca adequadamente o feixe de elétrons do tubo de raios catódicos (no caso de varredura aleatória : terminais vetoriais) ou gera uma matriz que contém informação sobre a intensidade adequada do feixe de elétrons para cada ponto da tela (no caso de varredura fixa : terminais raster).

A geração dos dados de entrada pode ser efetuada por dispositivos de entrada como descrito abaixo :

- JOYSTICK, CONTROL BALL, MOUSE : são usados para controlar os movimentos de um cursor sobre a tela do terminal. Baseiam-se quase sempre em potenciômetros ou transdutores de pressão que representam, através de tensões elétricas, as

coordenadas do ponto. São primitivamente dispositivos de posicionamento.

- TABLET : superfície plana sobre a qual, através do uso de um estilete é possível traçar ou digitalizar figuras, sendo a posição do estilete amostrada e as coordenadas do ponto indicado transmitidas ao sistema. O mecanismo de funcionamento dos tablet's varia bastante. Pode-se ter um campo eletrostático, magnético ou acústico na superfície e o estilete atua gerando uma perturbação no campo, que é traduzida em distância aos dois eixos x e y.

- LIGHT-PEN : é o único dispositivo que permite o apontamento direto de figuras, estando seu uso limitado aos terminais que necessitam de regeneração da imagem pois, só quando a regeneração acontece é possível determinar a figura apontada, através da sensibilização da celula fotoelétrica da light-pen (uma caneta que é apontada para a tela) pelo feixe de elétrons.

Existem ainda os teclados alfanúmericos para a digitação de letras, números e caracteres especiais e os teclados de funções programadas, cujas teclas, quando acionadas, determinam a execução de uma função pelo sistema.

O uso consagrou a classificação dos dados de entrada em cinco grupos (/OHLS 78/) :

- POSIÇÃO : coordenadas de um ponto na tela;

- VALOR : valores números;

- TEXTO : cadeia de caracteres alfanuméricos e/ou especiais;

- IDENTIFICAÇÃO DE OBJETO : um dado que representa a seleção de um objeto pertencente a uma figura mais complexa;

- IDENTIFICAÇÃO DE FUNÇÃO : um dado que representa uma função selecionada para ser executada pelo sistema.

Para que haja independência entre o dispositivo físico em uso e o programa que dele recebe informação, criou-se o conceito de dispositivo lógico ou virtual (/FOLE 74/, /OHLS 78/). Então, qualquer informação de entrada é considerada como fornecida por um dispositivo de uma das cinco classes lógicas, correspondentes aos cinco tipos de informação citados acima. Desta forma, mesmo que não se tenha em mãos o dispositivo clássico para uma destas classes (como é o teclado alfanumérico para a classe de TEXTO e a light-pen para a de IDENTIFICAÇÃO DE OBJETO), pode ser utilizado outro dispositivo físico através de sua simulação por software. Sem muito rigor, pode-se afirmar que um dispositivo físico de uma determinada classe pode simular qualquer das outras classes, através de software apropriado. Isto mostra-se por vezes conveniente, seja pela falta do dispositivo, seja para manter a continuidade tátil (evitar ao usuário operar

mais de um dispositivo).

Os primeiros programas dos sistemas de Computação Gráfica eram totalmente dependentes dos dispositivos gráficos utilizados. Assim, o programador necessitava conhecer características específicas do dispositivo e o conjunto de instruções, em linguagem de máquina, através das quais se gerava figuras, para poder programar as rotinas de entrada/saída da aplicação de acordo. Isto não trazia maiores problemas pois o uso de dispositivos gráficos era reduzido e somente efetuado por especialistas, já que havia poucos tipos de periféricos gráficos com baixa disponibilidade no mercado e custo relativamente alto.

Com o desenvolvimento da tecnologia e consequente queda de custo, os dispositivos gráficos foram se tornando mais populares e diversificados. Dois tipos de portatilidade assumiram então, importância, segundo Newman (/NEWM 78/), : de programas, traduzida como a habilidade de se transportar aplicações gráficas de uma instalação para outra sem necessidade de mudanças profundas nos programas, e de programadores, definida como a necessidade de se treinar programadores para entender as complexidades e peculiaridades de cada nova instalação gráfica.

Uma forma de atender aos dois tipos de portatilidade seria a padronização do hardware, meta muito difícil de se atingir devido à grande quantidade de fabricantes e aos diferentes estágios de desenvolvimento tecnológico que cada um deles possuía. Mesmo que se atingisse esta padronização, ainda seria difícil evitar que se desenvolvessem formas diferentes de utilização de um mesmo hardware.

Por outro lado, tornava-se cada vez mais necessária a existência de uma ferramenta para utilização em alto nível das capacidades gráficas dos dispositivos. Tal ferramenta evitaria que os programadores fossem obrigados à tediosa tarefa de preocupar-se com a linguagem de máquina dos dispositivos, em cada programa realizado.

O problema da interface, em alto nível, à aplicação foi resolvido com a criação dos chamados Pacotes Gráficos, constituídos por conjuntos de rotinas específicas para a geração e manipulação de figuras, normalmente desenvolvidos para dispositivos específicos e pelos próprios fabricantes. Por não existir nenhuma padronização, cada pacote tinha a sua filosofia de projeto e programas que utilizavam um determinado pacote muitas vezes necessitavam de profundas alterações se ocorresse a substituição de um dispositivo (e de seu pacote gráfico correspondente) por outro.

A figura 2.11 apresenta uma visão dos sistemas gráficos utilizando pacotes gráficos, onde cada pacote gráfico foi desenvolvido para um dispositivo específico (os chamados pacotes gráficos dependentes do dispositivo) e apresenta uma interface de programação à aplicação (ou seja, cada um dos pacotes é acessado pela aplicação de forma distinta). Em outras palavras, cada

pacote gráfico oferecia uma determinada linguagem gráfica procedural à aplicação, com primitivas próprias.

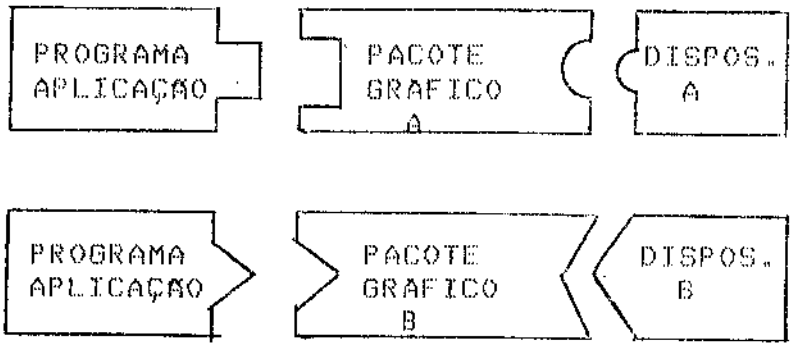


fig. 2.11 : Diferentes pacotes gráficos dependentes do dispositivo.

O problema de se obter os dois tipos de portatilidade, de programas e de programadores, continuou sem solução e cresceu de importância. A portatilidade de programas gráficos entre computadores de fabricantes diferentes e a independência destes programas em relação aos pacotes gráficos utilizados pelos dispositivos terminais também começaram a assumir importância, entre outras razões, pelo desejo de troca de informação entre a comunidade de usuários.

Em um próximo passo, caminhou-se para uma padronização da interface ao programador de aplicação, para que um programa de aplicação pudesse trabalhar com pacotes gráficos diferentes sem sofrer alterações. Isto significou padronizar a linguagem gráfica procedural a ser oferecida pelos diversos pacotes gráficos às aplicações. Esta nova situação é mostrada na figura 2.12.

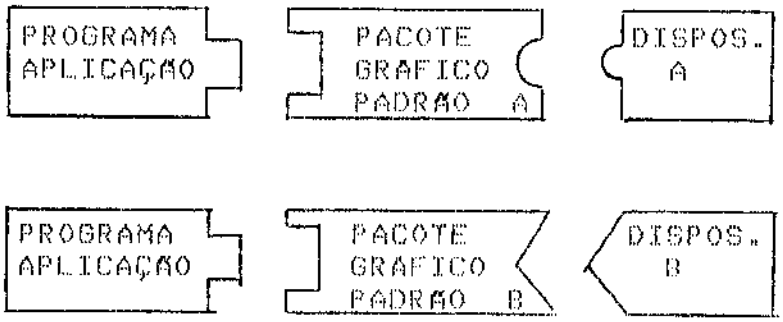


fig. 2.12 : Pacotes gráficos padronizados em relação à aplicação.

A seguir, surgiu a idéia do pacote gráfico tratar os dispositivos físicos como dispositivos lógicos, através do uso de driver's, ou seja, interfaces de entrada/saída uniformes que sejam padronizadas em relação ao pacote gráfico (sendo então acessadas de forma padronizada) e somente dependentes dos dispositivos. Tal tratamento permite abstrair para o pacote as similaridades entre os dispositivos e esconder nos driver's suas diferenças, beneficiando totalmente o programador de aplicação. Esta idéia foi pioneira no pacote GINO 3D (/WOOD 71/) e adotada posteriormente em outros pacotes (/BOS 77/). A figura 2.13 apresenta o diagrama desta última estrutura.

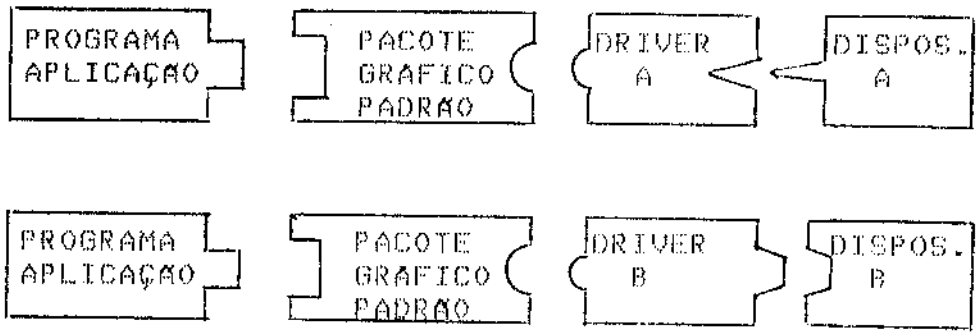


fig. 2.13 : Utilização de driver's.

Esta solução representa a tentativa de se obter uma interface padrão de alto nível entre o programa de aplicação e os dispositivos : o Núcleo Gráfico, um conjunto padrão de subrotinas. O Núcleo Gráfico deve oferecer ao programador de aplicação um conjunto de funções para usar dentro de seu programa, independentemente das características específicas dos dispositivos. Este conjunto de funções gráficas deve ser acessável em qualquer instalação gráfica, através dos driver's também padronizados.

Para se chegar na prática à configuração apresentada na figura 2.13, muito trabalho teve de ser desenvolvido (/NEWM 78/, /STRA 82/, /ENDE 84b/). Baseado em alguns sistemas existentes, como por exemplo, GINO-F (/GINO 75/), GRAFSY, GPGS (/BOS 77/), ESRO-UGP, PHILDIG (/PHIL 75/), CEFE, EZS, SKETCHPAD (/SUTH 63/), iniciou-se em 1975 uma discussão para a obtenção de um padrão gráfico único. Desta discussão, originaram-se basicamente duas propostas, o CORE SYSTEM nos Estados Unidos e o GRAPHICAL KERNEL SYSTEM (GKS) na Alemanha Federal. Ambas as propostas pretendiam a obtenção de um núcleo gráfico básico padrão (KERNEL ou CORE), independente dos dispositivos de entrada e saída, do hospedeiro, da linguagem e da aplicação.

As propostas se desenvolveram quase paralelamente e tiveram por requisitos as necessidades de englobarem todas as

capacidades essenciais a todo o espectro de aplicações gráficas e de controlarem toda a gama de dispositivos gráficos de maneira uniforme. De importância fundamental foi a definição de quais as funções essenciais a um pacote padrão : primitivas gráficas de saída, primitivas gráficas de entrada, funções para segmentação de figuras, funções de transformações de coordenadas, funções de controle. Outros pontos valiosos foram a identificação da necessidade de se estudar a estrutura de programas de aplicação para se entender como projetar sistemas de software para suportá-los e da vantagem de separar as funções essenciais para a geração de figuras (o CORE ou KERNEL) de outras funções, tais como de modelamento, mais específicas da aplicação.

Naturalmente, as propostas caminharam por caminhos um pouco diferentes, já que reuniam dentro de si idéias e experiências diferentes. Assim, a proposta CORE adotou sistemas de coordenadas tridimensionais, ao contrário das bidimensionais do GKS, o que tornou o pacote mais complexo e trouxe mais dificuldades em se obter um denominador comum.

Em 1975 foi publicado o primeiro relatório de andamento dos trabalhos do comitê de planejamento do software gráfico padrão (GRAPHICS STANDARDS PLANNING COMMITEE-GSPC) da ASSOCIATION FOR COMPUTER MACHINERY (ACM), apresentando uma pré-proposta do CORE SYSTEM.

Em 1977 foi publicada a primeira versão completa do que seria a norma CORE, tratando contudo somente vector graphics, ou seja, varrendo somente os dispositivos que utilizam vetores como unidade básica de desenho. (/GSPC 77/)

Em 1979, após o surgimento do GKS e em decorrencia de muitas discussões e análises do relatório de 1977, e de desenvolvimentos na área da computação gráfica, foi publicado um refinamento do CORE (/GSPC 79/). Esta versão integrou capacidades de sistemas raster às funções descritas na versão de 1977, pois o uso de hardware raster estava crescendo rapidamente. Assim, tanto dispositivos baseados em vetores quanto dispositivos baseados em matrizes de pontos poderiam ser controlados pelo CORE.

O GSPC foi dissolvido em 1979 e foi criado um comitê do AMERICAN NATIONAL STANDARDS INSTITUTE (ANSI), o órgão de padronizações dos EUA, denominado X3H3, que recebeu a missão de trabalhar o CORE para que pudesse ser adotado como padrão americano.

Por outro lado, o orgão oficial de padronização na Alemanha Federal, o DIN (DEUTCHES INSTITUTE FUER NORMUNG) desenvolveu e apresentou o GKS, uma proposta bidimensional que logo foi adotada naquele país.

Devido ao fato do GKS ter sido desenvolvido por uma instituição governamental e de assumir sistemas de coordenadas bidimensionais, o GKS pôde ser apresentado de forma concisa e completa em menor espaço de tempo do que o CORE. Assim, a

INTERNATIONAL STANDARDS ORGANIZATION (ISO) iniciou um trabalho no sentido de adotar o GKS como padrão internacional bidimensional, enquanto a ANSI continuava a trabalhar no CORE como proposta de padrão tridimensional.

Outros grupos de trabalho do ANSI apresentaram ressalvas ao GKS. Assim, em 1982, a norma GKS foi modificada (versão 7.2) para atender as principais restrições que lhe eram feitas pela comunidade internacional e foi então aceita como padrão internacional pela ISO. Isto esvaziou o trabalho no CORE, desviando os esforços para outras áreas, tais como a criação de padrões para :

- formatos de bases de dados que permitam intercâmbio de dados de projetos entre sistemas gráficos (IGES : INITIAL GRAPHICS EXCHANGE STANDARD). O propósito da norma é facilitar a comunicação, de dados de definição de produtos, entre sistemas CAD/CAM (Projeto e Fabricação auxiliados por computador) com diferenças de hardware, software de operação e software de aplicação. O âmbito inclui sistemas de projeto e desenho, sistemas de bases de dados, dispositivos de entrada/saída e programas de aplicação. O objetivo da norma é atingido através de uma estrutura de arquivo padronizada, decorrente de práticas existentes e de experiência coletiva, e de formatos de linguagem baseados em avançadas tecnologias de representação. Assim, a norma especifica uma estrutura de arquivo e formatos de linguagem para comunicação de dados de definição de produto, criados e utilizados por sistemas CAD/CAM. Entende-se por dado de definição de produtos aquelas informações, independentes de forma, re , ridas para descrever e comunicar características de engenharia de objetos físicos como produtos manufaturados. Estas informações incluem informação geométrica, topológica e não geométrica, requerida para desenhos de engenharia e modelos de produtos; (/IGES 81/)

- interface entre os sistemas gráficos e os dispositivos para que os driver's de dispositivos diferentes possam ser acionados de forma padronizada. Inicialmente tratada por VDI (VIRTUAL DEVICE INTERFACE), a norma CGI (COMPUTER GRAPHICS INTERFACE) reúne um conjunto de elementos básicos para o controle e o intercâmbio de dados entre software gráfico independente de dispositivo e um ou mais driver's dependentes dos dispositivos gráficos, onde driver é visto como a parte do sistema gráfico que traduz comandos e dados do CGI na forma requerida por um meio particular de entrada/saída. A norma CGI :

  - permite pacotes gráficos independentes de dispositivo interfaciarem-se, de forma consistente, com grande variedade de dispositivos gráficos ;

  - auxilia o intercâmbio de software entre instalações, isolando os aspectos que sejam dependentes de dispositivos;

- orienta fabricantes de dispositivos com relação a capacidades gráficas úteis.

O objetivo do CGI é, então, normalizar a interface entre pacotes gráficos e os dispositivos, do lado do pacote gráfico. Assim, mesmo que diversos pacotes tenham diferenças conceituais, podem acessar os driver's dos dispositivos de forma padronizada; (/VDI 84/, /ARNO 84/)

- armazenamento e transporte de figuras entre sistemas gráficos. Esta norma, inicialmente denominada VDM (VIRTUAL DEVICE METAFILE), e tratada hoje por CGM (COMPUTER GRAPHICS METAFILE), estabelece a metodologia, elementos funcionais e codificação de um metafile. O metafile é um mecanismo para reter e/ou transportar dados gráficos e informação de controle e contém uma descrição de figuras de modo independente do dispositivo. A norma CGM define um conjunto único utilizável de elementos de metafile para atender às seguintes necessidades :

- prover um metafile padrão para implementadores de pacotes gráficos;

- prover um padrão de transferência de figuras entre dispositivos, instalações e sistemas gráficos;

- prover um mecanismo de escape gráfico padrão para acessar capacidades não padronizadas de dispositivos gráficos através do metafile.

O CGM apresenta-se como uma interface de dados gráficos e relaciona-se em capacidade funcional ao CGI. A capacidade mínima do CGM inclui todos os elementos necessários para descrever figuras independentemente umas das outras. Estas figuras são armazenáveis em diferentes meios, transportáveis entre diferentes sistemas gráficos e exibíveis em diferentes dispositivos gráficos. Figuras no CGM podem ser corretamente interpretadas com processamento sequencial desde o início do metafile ou com acesso randômico ao início da figura e acesso sequencial dentro dela. A norma permite também a adição de dados não gráficos, dependentes da aplicação. São também incluídos formatos de codificação. (/VDM 83/, /ARNO 84/)

Apesar da aceitação da versão 7.2 do GKS como padrão pela ISO, grupos de trabalho das entidades de padronização continuam buscando sistemas gráficos mais completos que possam ainda ser adotados como padrões. Exemplos são :

- GKS 3D, extensão do GKS para operar com coordenadas tridimensionais (/GKS3 84/, /ARNO 84/);

- PMIG (PROGRAMMER'S MINIMAL INTERFACE TO GRAPHICS),

inicialmente trabalhado para ser um subconjunto do CORE, acabou sendo revisto e apresentado como um nível diferente da norma GKS, não definido por ela (/STRA 82/);

- PHIGS (PROGRAMMER'S HIERARCHICAL INTERACTIVE GRAPHICS SYSTEM), que adota como base as mesmas funções do padrão GKS, acrescentando tratamento tridimensional, combinação de funções de modelamento com funções de visualização (permitindo definir, exibir e modificar objetos relacionados geometricamente) e manipulação dinâmica de entidades geométricas. Para se ter a separação entre a criação e a exibição de uma figura, o PHIGS estrutura figuras que podem conter, por sua vez, subfiguras, o que permite a organização hierárquica de segmentos. (/PHIG 84/, /HEWI 84/, /STEI 84/, /ARNO 84/).

A partir do advento do núcleo gráfico padrão, para se ter um sistema gráfico são escolhidos os dispositivos de entrada e saída e confeccionadas interfaces entre o núcleo gráfico e estes dispositivos, o que torna o núcleo capaz de acessá-los para obter deles ou gerar através deles informações gráficas. Qualquer programa, para utilizar este sistema, necessita então de uma interface através da qual possa ativar as rotinas do núcleo.

O núcleo gráfico é então independente dos dispositivos porque se eles forem substituídos, só as suas interfaces serão alteradas; e independente do hospedeiro porque estará codificado em uma linguagem de alto nível, padrão, aceita e utilizada por uma larga faixa de fabricantes de computadores; é ainda independente da aplicação pois muitas aplicações diferentes podem utilizar os mesmos recursos do núcleo gráfico, que são padronizados.

Além das vantagens acima aludidas, o uso de um núcleo gráfico permite a portatilidade do programa de aplicação quando de seu transporte entre instalações oferecendo núcleos gráficos seguindo o mesmo padrão. Deve-se observar que pequenas alterações podem ser necessárias, por exemplo com relação aos nomes das rotinas do núcleo na nova instalação devido ao fato de a norma não especificar como implementar cada rotina ou quais nomes devem ser dados a elas, mas sim qual deve ser o efeito de cada uma delas, quais devem ser os parâmetros de entrada e de saída, que testes de erros devem ser realizados, que mensagens de erro devem ser providas, etc. Entretanto, para resolver o problema do acesso às rotinas do núcleo gráfico (particularmente com relação ao GKS), estão sendo desenvolvidas propostas para padronização dos nomes das funções oferecidas pelo núcleo, da ordem, do nome e do tipo de dado de cada parâmetro destas funções, para evitar alterações nas chamadas caso haja migração do programa de aplicação em relação a implementações distintas da mesma norma gráfica (como exemplo, a interface para a linguagem FORTRAN publicada sob o código ISO TC97/SC5/WG2 N214, em novembro de 1983).

Para atingir a independência do dispositivo, o núcleo

gráfico utiliza o conceito de classes lógicas de entrada, ou seja, qualquer informação a ser fornecida ao núcleo gráfico o será através de um dispositivo operando em uma das classes lógicas citadas anteriormente, e o conceito de primitivas de saída, segundo o qual, qualquer informação de saída será composta por uma ou algumas de diversas primitivas gráficas básicas.

São características do GKS :

- a definição de seis tipos de primitivas de saída :

  - POLYLINE : conjunto de linhas conectadas, definidas por uma sequência de pontos;

  - POLYMARKER : símbolos de um determinado tipo, centrados em posições fornecidas ;

  - TEXT : cadeia de caracteres numa dada posição ;

  - FILL AREA: uma área poligonal que pode ter somente o contorno ou estar preenchida com uma cor uniforme ou com um padrão ou estilo pré-definido ;

  - CELL ARRAY : um arranjo de unidades elementares do display ("pixel's"), com cores individuais ;

  - GENERALIZED DRAWING PRIMITIVE (GDP) : uma primitiva definida pelo usuário, baseada em características específicas do dispositivo que não estejam previstas no núcleo gráfico, como por exemplo, desenho de círculos, splines, etc.

- a definição de primitivas para seis classes de entrada :

  - LOCATOR : posições ;

  - VALUATOR : valores ;

  - STRING : caracteres ;

  - CHOICE : identificação de função ;

  - PICK : identificação de objeto ;

  - STROKE : conjunto de LOCATOR's.

- as características estáticas e dinâmicas das primitivas (tipo de linha, espessura de linha, tamanho de marcador, cor ou intensidade, etc) e dos dispositivos físicos são armazenadas em tabelas (estáticas) e listas (dinâmicas) que são preenchidas, alteradas ou consultadas sempre que um dispositivo de entrada é utilizado ou uma primitiva de saída é gerada ou manipulada. Assim, o GKS não se preocupa em como

um determinado dispositivo gera, por exemplo, um conjunto de retas e sim em fornecer informações suficientes para que qualquer tipo de dispositivo possa fazê-lo adequadamente. Também as características dependentes de implementação são armazenadas em uma tabela para proporcionarem subsídios aos recursos gráficos.

- qualquer dispositivo de entrada pode operar através do GKS segundo qualquer um de três modos operacionais :

  - REQUEST : o operador manipula o dispositivo de entrada quando explicitamente solicitado pelo programa de aplicação, que fica aguardando a entrada do dado;

  - SAMPLE : o programa de aplicação amostra o dispositivo de entrada e obtém dele o seu valor corrente;

  - EVENT : manipulações dos dispositivos de entrada geram eventos que são armazenados cronologicamente numa fila, a fila de eventos, acessável pelo programa de aplicação.

- o conceito de estação de trabalho (WORKSTATION) que é a forma pela qual o GKS enxerga qualquer dispositivo de entrada e/ou de saída. São definidas funções de controle que, quando ativadas pelo programa de aplicação, controlam as estações de trabalho e os estados operacionais do núcleo gráfico : OPEN GKS inicializa o sistema, OPEN WORKSTATION habilita a estação a realizar entrada, ACTIVATE WORKSTATION habilita a estação a realizar saída, DEACTIVATE WORKSTATION, CLOSE WORKSTATION e CLOSE GKS, desabilitam, respectivamente, saída, entrada e o sistema gráfico. Funções de controle adicionais permitem limpar tela, redesenhar os segmentos, enviar mensagens ao operador, etc.;

- permitir o agrupamento de primitivas de saída em segmentos, manipuláveis como um conjunto (segmentos podem ser criados, deletados, renomeados, transformados) e possuindo características específicas adicionais (detectabilidade, cintilamento, visibilidade, prioridade), que podem ser dinamicamente alteradas, provocando modificação visual dos segmentos;

- definir três espaços de coordenadas (mundo, interno normalizado e do dispositivo), as transformações entre eles e o "clipping" (recorte das partes da figura que excedam os limites da janela de exibição escolhida), permitindo flexibilidade no tratamento das figuras desde sua definição pela aplicação até a exibição na tela do terminal. As figuras são definidas em coordenadas mundo (da aplicação). Uma window especifica uma porção de interesse neste espaco enquanto uma viewport especifica uma janela no espaço normalizado interno. A transformação de normalização

executa o mapeamento da window na viewport. Analogamente, uma workstation window especifica uma porção de interesse no espaço normalizado interno que será mapeada pela transformação de workstation sobre uma janela na tela do terminal, a workstation viewport. Existe uma transformação de workstation para cada estação de trabalho ativa;

- permitir modificação dinâmica de figuras e o controle do momento em que as modificações devem provocar efeito visual: até um pedido de regeneração explícita, até a próxima interação na estação de trabalho específica ou até a primeira interação global (em qualquer estação de trabalho);

- permitir armazenamento temporário de segmentos (SEGMENT STORAGE) e permanente de segmentos, primitivas exteriores a segmentos e dados não gráficos (METAFILE);

- formalizar estados operacionais para o núcleo gráfico (GKS fechado, GKS aberto, ao menos uma estação de trabalho aberta, ao menos uma estação de trabalho ativa, segmento aberto); cada função da norma tem especificados os estados operacionais nos quais pode ser ativada;

- definir todas as situações possíveis de ocorrência de erros, especificando as mensagens que devem ser enviadas ao usuário;

- apresentar níveis implementáveis independentemente, com capacidades específicas, tornando possível a utilização daquele nível que melhor atenda às necessidades de instalações e/ou aplicações específicas. A versão 7.2 do GKS especifica nove níveis de 0A a 2C.

Em /GKS 82/, /ENDE 84b/, /BONO 82/ e /SILV 85a/, podem ser encontradas informações detalhadas sobre a norma para software gráfico, GKS.

O CORE SYSTEM, por sua vez, apresenta às seguintes características, em sua versão de 1977 :

- figuras são definidas em coordenadas mundo, bi ou tridimensionais, através de chamadas às funções primitivas de saída. Cada primitiva é definida pelos parâmetros da função (coordenadas, caracteres ou tipo de marcador), a transformação de visualização e os valores correntes dos atributos da primitiva. Nenhum aspecto da primitiva pode ser alterado após a sua criação;

- atributos : estilo e espessura de linha, intensidade, cor, fonte, tamanho, espaçamento, qualidade e plano de caracteres, e identificador de pick;

- transformações de visualização especificam uma porção de interesse no mundo gráfico a ser exibida em uma área

específica da tela do dispositivo. São dois sistemas de coordenadas : o sistema de coordenadas mundo e o sistema de coordenadas normalizado do dispositivo. A transformação de visualização pode ser bi ou tridimensional, dependendo da dimensionalidade das coordenadas mundo. No primeiro caso, a transformação é totalmente especificada por um retângulo (inclinado ou não) no espaço de coordenadas mundo, a window, e um retângulo no plano de coordenadas normalizado (paralelo aos eixos), a viewport. O clipping (recorte) é feito obrigatoriamente, nos limites da window, ou seja, as partes da figura que estão fora dos limites da window são sempre perdidas. Em três dimensoes, é necessário especificar, além de window e viewport, uma projeção ao plano bi-dimensional, através de um plano de projeção dentro do mundo gráfico e, ou um centro de projeção (para projeções perspectivas) ou uma direção de projeção (para projeções paralelas). Na transformação de visualização tridimensional, dois clipping's ocorrem : um de profundidade que elimina as figuras abaixo do plano de projeção e outro que elimina objetos cujas projeções ficam além dos limites da window;

- qualquer primitiva de saída deve estar contida em um segmento. O segmento é aberto, funções pr...tivas são acionadas e o segmento é fechado. A cada instante, só um segmento pode estar aberto e depois que o segmento é fechado não podem ser acrescentadas primitivas a ele. Existem dois tipos de segmentos : retidos e não retidos. Os não retidos são criados quando não se pretende utilizar o segmento mais de uma vez, nem alterar seus atributos; são automaticamente deletados quando ocorre uma regeneração da imagem. Os retidos permitem alteração dinâmica de seus atributos de visibilidade, cintilamento, detectabilidade e de transformação de coordenadas (concatenação de translação, escalamento e rotação na superfície de exibição). O atributo de primitiva chamado identificador de pick permite sub agrupamento de primitivas dentro de um segmento, para utilização na entrada gráfica. Segmentos retidos recebem nomes dados pela aplicação e podem ser explicitamente deletados ou renomeados;

- o CORE enxerga os dispositivos de entrada gráfica como sendo dispositivos lógicos, baseado no conceito de classes lógicas de entrada já referido anteriormente. As classes recebem os nomes de :

  - PICK : uma primitiva de saída;

  - LOCATOR : uma posição na superfície de exibição;

  - VALUATOR : um valor escalar;

  - KEYBOARD : uma cadeia de caracteres;

  - BUTTON : uma chave pré-programada.

Cada dispositivo deve ser habilitado para ser utilizado e tem associado um tipo de realimentação ao operador que pode ser ligado ou desligado pela aplicação. Existem dois modos de entrada : EVENT, através do qual dispositivos das classes PICK, KEYBOARD e BUTTON geram eventos que são cronologicamente registrados numa fila de eventos acessável pela aplicação, e SAMPLE, através do qual dispositivos das classes LOCATOR e VALUATOR podem ser amostrados pelo programa de aplicação que obtém então seus valores correntes. São permitidas associações de dispositivos do modo SAMPLE a dispositivos do modo EVENT de forma que, quando ocorre um evento, os dispositivos SAMPLE associados ao dispositivo que gerou o evento são amostrados e seus valores correntes registrados no relatório do evento;

- com relação ao controle, dentre outras facilidades, o CORE permite :

  - uso de múltiplas superfícies de exibição que devem ser inicializadas e podem ser dinamicamente selecionadas ou deselecionadas;

  - inquire de status e variáveis do sistema e de propriedades dos dispositivos;

  - adiamento do efeito visual das mudanças nas figuras sendo exibidas : ou as modificações são automáticas ou são armazenadas para serem realizadas de uma só vez;

- foram definidos quatro níveis válidos de implementações CORE, onde cada nível engloba as capacidades dos níveis anteriores :

  - NIVEL 1 : saída básica com atributos em segmentos não retidos, transformações de visualização e funções de controle;

  - NIVEL 2 : segmentos retidos e modificações de atributos de visibilidade e cintilamento;

  - NIVEL 3 : modificação do atributo de detectabilidade e capacidades de entrada;

  - NIVEL 4 : transformações de coordenadas dos segmentos.

- o CORE permite o uso do conceito de posição corrente : um valor que define a localização atual de desenho, dentro das coordenadas mundo. Para cada segmento, a posição corrente é inicializada em zero, uma posição inicial deve ser dada e as primitivas subsequentes têm suas coordenadas especificadas relativamente à posição corrente. Assim, cada primitiva de saída se subdivide em duas, uma com coordenadas absolutas e outra com coordenadas relativas.

Em /GSPC 77/, /GSPC 79/, /MICH 78a/, /MICH 78b/, /BERG 78/, podem ser obtidos mais detalhes sobre o CORE SYSTEM.

Comparando as duas propostas, a de 1977 (com o refinamento raster de 1979) para o CORE e a de 1982 para o GKS, nota-se que :

- as primitivas de entrada e de saída se equivalem. O CORE apresenta a facilidade de se definir coordenadas relativas para as primitivas de saída, permitindo a existencia do conceito de posição atual (uma reta ser traçada da posição atual até uma determinada posição, por ex.);

- ao contrário do GKS, no CORE não existem estações de trabalho, sendo os dispositivos de saída considerados como terminais e os dispositivos de entrada sendo considerados sempre independentes dos de saída;

- no CORE existem somente dois espaços de coordenadas, um para a aplicação e outro normalizado, mas ambos com três eixos de coordenadas;

- no CORE, as características das primitivas de saída são sempre estáticas, ou seja, depois de definidas não podem ser alteradas; com relação aos atributos, o GKS além de permitir sua modificação dinâmica permite a definição de conjuntos de atributos (BUNDLES) que são associados a índices os quais são utilizados na geração das primitivas;

- a definição de níveis de implementação está presente em ambas as propostas, com o mesmo objetivo;

- ao contrário da norma GKS, o CORE talvez por estar ainda em desenvolvimento, quase não contém normas de implementação, o que certamente faz com que implementações diferentes tragam comportamentos diferentes;

- com relação à segmentação, o CORE não permite a existência de primitivas externas a segmentos, enquanto o GKS sim, mas o CORE permite dois tipos de segmentos : o manipulável (retido) e o não manipulável (não retido). O atributo de prioridade de segmentos existente no GKS não está definido no CORE;

- quanto aos modos operacionais de entrada, o GKS permite qualquer dispositivo em qualquer dos três modos; o CORE, por sua vez , não define o modo REQUEST e restringe o modo EVENT aos dispositivos das classes PICK, KEYBOARD e BUTTON e o modo SAMPLE aos das classes LOCATOR e VALUATOR. O CORE pode se equivaler ao GKS na operação de LOCATOR e VALUATOR como EVENT, já que permite associar dispositivos SAMPLE a dispositivos EVENT possibilitando registrar, nos relatórios de eventos, dados amostrados (SAMPLE);

- ao contrário dos três estados de adiamento existentes no GKS, o CORE só permite um;

- o CORE não apresenta a facilidade do GKS de não realizar obrigatoriamente o clip na window.

Em /ENCA 80/, podem ser obtidos mais detalhes sobre similaridades e diferenças entre o CORE e o GKS.

Não são conhecidos muitos trabalhos de implementação de sistemas gráficos baseados na norma CORE no Brasil, mas algumas instituições iniciaram trabalhos nesta direção antes que o GKS surgisse como mais provável proposta a ser aceita e hoje existem alguns sistemas baseados no CORE, como por exemplo o desenvolvido na UFRGS, intitulado PGE (PACOTE GRAFICO EXPERIMENTAL) (/FREI 84a/, /FREI 84b/). Internacionalmente, apresentam exemplos de implementações CORE : /FOLE 81/, /FREI 81/, /NICO 81/, /STLU 82/.

Em virtude do estágio atual das duas propostas de padronização, quando o CORE ainda merece cuidados e o GKS já se consolida como padrão internacional tendo baseadas em si diversas implementações em utilização por todo o mundo (conforme listas apresentadas por Enderle (/ENDE 83/ e /ENDE 84/), daqui por diante sempre se referirá a núcleo gráfico como aquele seguindo a norma GKS. Deve ser ressaltado, entretanto, que todo o formalismo a ser apresentado se mantém válido caso seja utilizado o padrão CORE, por estar o núcleo gráfico sempre tratado através de suas diversas interfaces, o que permite a sua substituição sem maiores transtornos.

2.4-CARACTERIZAÇAO DO USUARIO

O usuário representa, no diagrama apresentado na figura 2.2, a pessoa que opera o computador, através de um terminal, trocando informações com o programa em execução. Seu objetivo é a realização de uma determinada tarefa, utilizando um meio de comunicação para acessar os recursos computacionais alocados à aplicação. De modo geral, o operador é um elemento mais voltado para a aplicação, ou seja, uma pessoa que não tem conhecimentos profundos de computadores e periféricos e menos ainda de programas para computadores. Ele conhece a aplicação, sabe o que precisa fazer e como fazer. Bons programas de aplicação podem controlar uma aplicação satisfatoriamente mas não o desempenho de um operador. Assim, é vital para o sucesso do trabalho que o operador se sinta à vontade quando operando o sistema. Devem ser evitados atrasos desnecessários, perguntas dúbias, caminhos obscuros, enfim, qualquer tipo de situação incômoda ao operador que possa prejudicar sua atuação. Em resumo, o sistema deve ser "amigável" ("USER-FRIENDLINESS") ao usuário.

Em /DEHN 81/ define-se um sistema de diálogo amigável ao usuário, de um ponto de vista tecnológico, como sendo aquele onde o usuário individual sente muito baixa a complexidade operacional das atividades que tem de realizar para atingir a tarefa pretendida.

Hoje existem diversos estudos publicados a respeito do papel do operador e de suas características, inclusive psicológicas.

Em /HANU 82/ são identificados quatro fatores que influem no comportamento de operadores humanos :

- GRAU DE CONHECIMENTO : modelo que o operador tem do computador, em função da freqüência de utilização de computadores para resolver seus problemas;

- GRAU DE EXPERIENCIA : resultante da frequência com a qual o operador se utiliza de um sistema de aplicação específico e do nível de treinamento do operador para se comunicar com este sistema;

- PREFERENCIAS PESSOAIS : que métodos um operador prefere para comunicar-se com um programa de aplicação (linguagem de comandos, menus, etc.);

- "FISIONOMIA" : que tipos de dispositivos são preferivelmente usados por uma comunidade de operadores.

Já Martin (/MART 73/) identifica diversas categorias, não excludentes, de operadores e salienta que o autor de diálogo deve conhecer antecipadamente para que categorias de operadores

deve projetar cada diálogo ou, se o dialogo a projetar deverá atender a um espectro amplo de usuários. Pode-se citar operadores :

- DEDICADOS, que trabalham constantemente no terminal (estão treinados), ou CASUAIS, que utilizam o terminal ocasionalmente (não estão treinados);

- EXPERIENTES EM PROGRAMAÇÃO : são capazes de realizar ou de entender tarefas que aqueles sem experiência ou sem conhecimentos de programação não são;

- INTELIGENTES : possuem raciocínio lógico, memória de "rápido acesso" e capacidade de associação;

- ALTAMENTE TREINADOS : são mais capazes do que operadores sem treino ou pouco treinados;

- ATIVOS, que inicializam as operações no computador, fazendo perguntas, fornecendo dados ou inicializando o processamento, ou PASSIVOS, para os quais o computador toma a iniciativa dos diálogos;

- INTERMEDIARIOS : intermediam a comunicação entre usuários que desconhecem a máquina e o programa então em execução;

- que trabalham sob CONDIÇÕES NEGATIVAS (cansaço, desconforto, baixa luminosidade, etc.);

- que realizam MÚLTIPLAS APLICAÇÕES (mnemônicos de uma podem lembrar comandos de outra).

Por outro lado, Hollnagel (/HOLL 83/) lista uma série de questões não totalmente respondidas até hoje, das quais merecem destaque :

- O QUE CAUSA A SELEÇÃO DE UMA ESTRATEGIA ? - estratégias são decisões regulares planejadas que efetivamente trazem um resultado específico. Uma estratégia que mostra-se ótima para determinadas circunstâncias, entretanto, pode mostrar-se totalmente ineficaz sob outras circunstâncias. A prática tem mostrado que frequentemente operadores não utilizam a estratégia ótima para uma determinada situação e até hoje as causas disto não são totalmente conhecidas;

- O QUE CAUSA O CHAVEAMENTO ENTRE ESTRATEGIAS ? - outro problema que ocorre é o operador mudar de estratégia no meio de uma sessão de trabalho. Que isto ocorre pode ser identificado mas porque ocorre, não;

- COMO SÃO TOMADAS DECISÕES ? - para analisar a performance do operador é importante saber como ele toma suas decisões, principalmente durante situações críticas. O que o operador leva em conta para optar por uma determinada ação ?

- MUDANÇAS NO NIVEL DE COMPORTAMENTO - o comportamento do operador é descrito segundo três níveis : baseado na experiência, em regras ou em conhecimento. Comportamento baseado em experiência é caracterizado por seqüências de atividades imediatamente disponíveis que são aplicadas para atingir uma condição expecífica, sendo executadas sem requerer a atenção do operador. No comportamento baseado em regras, a seqüência de atividades é controlada por uma descrição disponível ao operador e é requerida sua atenção durante a execução. O comportamento baseado em conhecimento, por sua vez, refere-se à seqüência de atividades não guiadas por qualquer plano ou descrição mas sim produzida pela interpretação da situação pelo operador, exigindo dele total atenção.

Em /DEHN 81/ é dedicado um capítulo à apresentação e avaliação de extensa literatura a respeito do usuário e da "user-friendliness". Atenção é dada aos fatores que influem no comportamento do usuário e que devem ser considerados se o projeto do sistema de diálogo pretende ser feito orientado para o usuário. São exemplos de fatores influentes : tipos de necessidades, treinamento, características (paciência, tolerância, memorização, concentração), educação e/ou experiência em processamento de diálogos, tipo de área de aplicação na qual está envolvido, razões para utilizar o sistema, atitudes e expectativas em relação ao sistema, objetivos, estrutura e restrições de tempo da tarefa a realizar.

Outros trabalhos têm procurado analisar a alocação, ou distribuição, de tarefas entre o operador humano e o computador (/EASO 80/, /PATA 84/). Pataca (/PATA 84/) realizou um modelamento matemático da atuação do operador, com o objetivo de estudar o seu comportamento e avaliar a influência deste comportamento no desempenho global do sistema interativo.

Neste item são detalhadas as quatro interfaces identificadas nos sistemas interativos, conforme apresentado na figura 2.2.

2.5.1-Interface Aplicação/Diálogo

Esta Interface estabelece um protocolo de comunicação entre as camadas de aplicação e de diálogo, sendo definida em conjunto pelos projetistas de diálogo e de aplicação. A definição desta interface possibilita dividir-se a resolução de um determinado problema em duas partes, quais sejam a parte primordialmente interessada na área do problema de aplicação e a parte interessada na área de comunicação entre operadores humanos e computadores.

Isolando do programador de aplicação a necessidade de estudar técnicas de interação homem-máquina e teorias de fatores humanos, para projetar seus sistemas de aplicação, torna-se amena a sua tarefa, já que, podendo contar com a implementação da comunicação usuário-computador realizada por um especialista em interação, o programador de aplicação pode concentrar seus conhecimentos na área de aplicação. Por outro lado, o projetista de interação pode trabalhar em sequências de interação usuário-computador pré-definidas, independentemente do teor das áreas de aplicação nas quais as sequências serão utilizadas.

Em resumo, a definição da interface aplicação/diálogo permite que o programador de aplicação projete seus programas supondo as informações necessárias, do e para o usuário, sendo tratadas por mecanismos independentes da aplicação.

2.5.2-Interface Diálogo/Núcleo Gráfico

A Interface Diálogo/Núcleo Gráfico gerencia a utilização do Núcleo Gráfico pela camada de Diálogo. Em outras palavras, permite ao diálogo o uso de funções gráficas oferecidas pelo núcleo gráfico.

É responsabilidade desta interface englobar em si todas as dependências existentes entre as camadas de diálogo e de núcleo gráfico, de forma a torná-las conceitualmente independentes (portáteis). Isto é particularmente útil pois permite um projeto estruturado do sistema interativo através do uso de núcleos gráficos já existentes na instalação ou, pelo

Em decorrência desta responsabilidade, a interface diálogo/núcleo gráfico deve se ocupar de todas as conseqüências surgidas da decisão de se adotar um núcleo gráfico padrão para o apoio à geração e manipulação de figuras. Estas conseqüências, reunindo facilidades e dificuldades, são detalhadas no item 3.3 e se referem à necessidade de adequação do núcleo gráfico à camada de diálogo. Em termos de implementação, são definidas primitivas adequadas ao diálogo que são mapeadas pelo gerenciador aos recursos disponíveis no núcleo gráfico, após o devi. ratamento.

2.5.3-Interface Núcleo Gráfico/Dispositivos

Esta interface, também conhecida como Interface de Entrada/Saída, realiza a conversão de instruções independentes de dispositivo, geradas pelo núcleo gráfico, às capacidades gráficas dos dispositivos físicos utilizados. Reúne dentro de si todo o tratamento das peculiaridades de cada dispositivo, permitindo ao núcleo gráfico atuar sobre eles de forma padronizada.

Como apresentado em 2.3, o núcleo gráfico é independente dos dispositivos gráficos de entrada e saída utilizados. Assim, é necessário que algum dos componentes do sistema interativo se responsabilize pelo conhecimento das características de cada dispositivo físico e pela sua utilização por parte do núcleo gráfico. Este componente é a interface de entrada/saída (apresentada no item 2.3 na forma de driver's).

Quando o núcleo gráfico deseja desenhar uma primitiva de linha, por exemplo, ele aciona a interface que verifica qual o recurso de cada dispositivo de saída para a geração de retas e providencia a informação específica necessária a cada um deles. Por outro lado, quando o usuário aciona algum dispositivo de entrada, a interface é novamente acionada, agora para receber o dado fornecido e, de acordo com as características específicas daquele dispositivo, interpretar o dado e passá-lo ao núcleo gráfico.

Em resumo, a interface de E/S realiza uma correspondência entre capacidades do núcleo gráfico e dos dispositivos para a geração e o recebimento de informações gráficas, ressaltando que ao núcleo gráfico não interessa de que forma são gerados dados de saída ou fornecidos dados de entrada, mas sim de que tipo são estes dados e que informação eles realmente representam.

O padrão de software gráfico só normaliza a linguagem gráfica procedural a ser oferecida às aplicações (ou seja, a interface do núcleo gráfico para a aplicação). Assim, como se dá a tradução dos códigos independentes de dispositivo, gerados pelo núcleo gráfico, aos códigos dependentes de dispositivos, particulares de cada dispositivo, varia entre implementações

diferentes de um mesmo padrão gráfico. Logo, não existe portatilidade de driver's em relação a implementações do núcleo gráfico. Como mencionado no item 2.3, a norma CGI (COMPUTER GRAPHICS INTERFACE) (/VDI 84/) está sendo desenvolvida justamente com o objetivo de padronizar esta linguagem gráfica de saída do núcleo gráfico, ou seja, padronizar o acesso do núcleo gráfico aos driver's dos dispositivos. Assim, adotando-se o CGI, podem ser confeccionados driver's entre dispositivos conhecidos e implementações de núcleo gráfico desconhecidas, em virtude da independência, então atingida, do núcleo gráfico em relação ao driver, o que se traduz em portatilidade dos driver's em relação a implementações de núcleo gráfico.

Molinaro (/MOLI 81/) e Martins (/MART 81/) realizaram trabalhos na área de interfaces de entrada/saída e Silva (/SILV 85c/) descreve aspectos de projeto e implementação de tais interfaces, apresentando uma interface realizada entre uma implementação segundo a norma GKS versão 6.2 e o terminal gráfico vetorial regenerativo GT40.

2.5.4.-Interface Usuário/Sistema Interativo

Esta interface, mais conhecida na literatura simplesmente como Interface de Usuário, representa a visão que o usuário (entendido como sendo o operador do programa) tem do sistema (entendido como sendo aplicação mais diálogo) e de sua qualidade depende a compreensão do sistema pelo usuário e, consequentemente, o sucesso em sua operação. Uma boa interface torna programas não somente fáceis de aprender mas tambem eficientes para se operar. Por outro lado, uma interface de usuário mal projetada pode tornar o programa não operável.

É o projeto da estrutura do diálogo que define esta interface, ou seja, o projetista de diálogo é quem determina a forma pela qual o usuário enxerga o sistema interativo.

Diversos autores analisam a interface de usuário e descrevem suas caracterlsticas.

Em /DEHN 81/ é apresentada uma representação detalhada de conceitos relacionados ao diálogo homem-computador e são discutidos o fenômeno de diálogo e técnicas de diálogo. Todos estes conceitos e técnicas têm por objetivo a obtenção de interfaces de usuário voltadas para o usuário. Quatro classes de objetivos relevantes na interface de usuário podem ser citados como sendo a flexibilidade do diálogo, a transparência do comportamento do sistema em relação ao usuário, a facilidade de aprendizado e de uso por parte do usuário e a confiabilidade do diálogo.

Já Stewart (/STEW 80/), tendo em vista a capacidade de absorção de informação e de realização de tarefas, próprias de operadores humanos, aponta nove aspectos de importância no

projeto de diálogos entre operadores e programas :

- manter documentação ordenada, clara, atualizada, completa e utilizando vocabulário familiar à classe de pessoas envolvidas;

- permitir variações no grau de dificuldade do diálogo que possibilitem aos operadores novatos se inteirarem gradativamente das facilidades oferecidas;

- utilizar abreviaturas e códigos consistentes e coerentes entre si;

- dar tanta importância ao layout quanto ao formato da informação. São importantes :

  - sequência lógica de apresentação da informação em relação ao usuário e ao sistema;

  - agrupamento de itens similares melhora a legibilidade e relaciona os dados;

  - espaçamento adequado entre as informações;

  - somente apresentar informações relevantes;

  - os formatos na tela devem ser consistentes;

  - simplicidade nas informações;

- a sequência do diálogo deve corresponder à sequência de tarefas do usuário, ou seja, os passos a serem dados pelo operador devem corresponder à sequência lógica de atitudes dentro da tarefa de aplicação;

- recursos de computação gráfica normalmente permitem uma comunicação mais efetiva do que outros meios;

- minização de erros;

- tempo de resposta do sistema;

- testes com usuários reais antes de colocar o diálogo em uso normal.

Shackel (/SHAC 80/) também analisa o diálogo, apontando problemas e ressaltando a importância da existência de uma linguagem mutuamente compreendida pelos participantes do diálogo.

Newman (NEWM 81/) identificou quatro componentes na interface de usuário, os quais serão descritos a seguir.

O primeiro componente é o MODELO DO USUARIO, uma

abstração do sistema interativo, sob o ponto de vista do operador, ou seja, uma formalização do comportamento do sistema em relação ao operador. O modelo possibilita ao operador desenvolver um entendimento do que o programa está fazendo.

Para manipular seu modelo, o usuário necessita de comandos. O conjunto de comandos colocado à disposição do usuário forma a LINGUAGEM DE COMANDO, identificada como o segundo componente.

O terceiro componente é a EXIBIÇÃO DE INFORMAÇÃO, que permite expor ao usuário a informação gerada.

Um último componente é a REALIMENTAÇÃO. Provida pelo computador para auxiliar o usuário na operação do programa, a realimentação serve para informá-lo de que seus comandos foram adequadamente recebidos e entendidos pelo programa.

Na sequência, cada um destes quatro componentes é detalhado.

2.5.4.1-Modelo do Usuário

O objetivo na definição de um modelo do sistema sob o ponto de vista do usuário é tornar este sistema totalmente inteligível ao usuário o que beneficia a sua operação, pois através do modelo, o usuário visualiza o efeito de cada ação permitida e pode planejar suas próprias estratégias para operar o programa.

O modelo deve empregar conceitos familiares ao usuário, para ser intuitivo e de fácil aprendizagem. Em uma situação ideal, a obtenção do modelo vem de uma simulação do próprio mundo real do usuário. Mas esta simulação normalmente traz muitas dificuldades para ser realizada.

O modelo pode ser representado como um conjunto de OBJETOS e um conjunto de AÇÕES que o usuário pode aplicar aos objetos. Cada objeto é um item de informação sobre o qual o usuário tem algum controle: ou simplesmente exibí-lo ou modificá-lo e destruí-lo. Estes OBJETOS podem ser de dois tipos : os INTRINSECOS, que são específicos da aplicação e sobre os quais normalmente o usuário tem controle total, e os de CONTROLE, que auxiliam na operação do programa, normalmente controlados pelo próprio programa (cursores, menus de comandos, seleções, escalas, diretrizes, grades, etc.). Nem todas as AÇÕES possíveis podem ser aplicadas a todos os objetos e o modelo deve espelhar quais ações são válidas para cada objeto.

2.5.4.2-Linguagem de Comando

A Linguagem de Comando reúne todos os comandos necessários à operação do programa, não devendo ser visualizada como uma simples listagem de comandos mas como uma linguagem, já que os comandos estão relacionados entre si através de uma sintaxe e cada construção efetuada com um ou mais comandos e seus possíveis operandos tem um significado semântico.

Uma análise mais profunda revela que existe uma estreita correspondência entre os OBJETOS e as AÇÕES do modelo do usuário e os COMANDOS e seus OPERANDOS da linguagem de comando. Mas, nem sempre esta correspondência se faz um a um já que podem ser necessários vários comandos para a execução de uma única ação mais complexa.

São pontos importantes a serem considerados no projeto da linguagem de comando :

- MODOS DE COMANDO : se uma mesma ação do usuário pode ser interpretada de formas diferentes, dependendo do estado do programa, cada estado é um MODO. Linguagens uni-modais são preferíveis para evitar enganos;

- SEQUENCIA DE SELEÇAO : o operando pode ser selecionado antes ou depois de selecionada a operação. No segundo caso, estar-se-ão introduzindo modos de comando adicionais pois, após um comando DELETE, um comando de SELEÇAO seleciona somente um item (o item a ser deletado), enquanto após um comando MOVA, o comando de SELEÇAO implica na seleção de dois itens (o item a ser movido e a nova posição). Então, linguagens unimodais pressupõem seleção prévia dos operandos;

- MECANISMO DE ABORTO DE COMANDOS : para comandos envolvendo sequências de passos, deve ser permitido ao usuário voltar atrás durante a seleção;

- MECANISMO PARA MANIPULAÇAO DE ERROS : devem ser decididas as formas de tratamento de entradas erradas ou sem significado.

Alguns estilos mais comuns de linguagens de comando :

- DIALOGO DE TECLADO SIMPLES : o usuário digita respostas no teclado alfanumérico às perguntas feitas pelo computador. A faixa de respostas aceitáveis é limitada e pode vir entre parênteses, após a pergunta, para facilitar ao usuário. Permite diálogos complexos mas é ineficiente em relação às linguagens gráficas pois frequentemente necessita de grandes sequências de respostas;

- LINGUAGEM DE COMANDO DE TECLADO : é um aprimoramento do estilo anterior, onde uma mensagem contém a informação de

uma sequência completa de respostas. Em uma sintaxe simples, usam-se quatro elementos na mensagem : um verbo (AÇÃO), operandos (OBJETOS), modificadores (alteram a interpretação dos operandos) e delimitadores (espaço, "return"). Ex : DELETE 1 ATE 100. São indicadas quando o teclado é o único dispositivo de entrada ou quando são necessários muitos itens alfanuméricos;

- CHAVES DE FUNÇÃO : os dados são fornecidos por entrada gráfica mas os comandos através de chaves de função programadas (DELETE, MOVA, etc.). Comparada com a seleção por menu, apresenta as vantagens de rapidez na seleção, não ocupação de espaço na tela e de facilidade de assimilação pelo operador. Na falta de chaves programadas, chaves comuns podem simulá-las, operando em outro modo de comando;

- LINGUAGEM DE COMANDO POR MENU : os menus são apresentados verticalmente num dos lados da tela ou horizontalmente acima ou abaixo e contêm todas as opções de escolha permitidas, no momento, ao usuário, que pode então escolher um OBJETO da aplicação, uma AÇÃO a ser executada ou o modo de operação do programa (EDIÇÃO, ANÁLISE, etc.). Os elementos de cada menu são determinados pelo modelo do usuário. São de importância no projeto dos menus :

  - o espaço ocupado na tela, já que espaço é fator crítico. Se a quantidade de itens for grande, deve-se usar menus multi-níveis. Símbolos gráficos ao invés de texto é uma boa opção;

  - a decisão entre seleção prévia ou não dos operandos, conforme já foi visto;

  - o instrumento de seleção; com a light-pen o item selecionado é diretamente identificado; com outros dispositivos, as coordenadas devem ser obtidas.

Menus permitem ao usuário visualizar a faixa de alternativas possíveis, protegem contra escolhas inválidas, já que só as válidas estão no menu, são fáceis de construir e são frequentemente representados como segmentos separados de "display-file", permitindo chaveamento por "posting" e "unposting" (forma de colocar ou retirar o conjunto de instruções que geram aquele segmento na tela, do ciclo de "refresh" do processador do terminal, através da mudança de dois apontadores);

- PROGRAMAS "PAINTING" : úteis para aplicações do tipo de layout de tubulações ou fiações, simples desenhos de linhas ortogonais e arte para animação. Normalmente será oferecido ao usuário um conjunto de "pincéis" para uso na aplicação de "pintura" ("painting") da imagem na tela. Caso exista capacidade de cor, deve existir um menu com as tonalidades permitidas, senão a opção entre preto e branco é dada ao usuário. Após selecionar o "pincel" e a cor, o usuário

aponta os locais ou traça riscos. Podem ser incluídos comandos na mesma sintaxe, em "pincéis" especiais;

- RECONHECIMENTO DE CARACTERES "ON-LINE" : utiliza-se uma rotina para reconhecer caracteres ou símbolos, feitos à mão livre pelo usuário, comparando-os com matrizes em um dicionário. Em caso de sucesso, a rotina devolve identidade, tamanho e posição do caracter. Apesar de vantagens quanto à grande quantidade de informação que pode ser extraída de cada ação do usuário e quanto à habilidade de se poder treinar o reconhecedor para aceitar símbolos desenhados da maneira mais conveniente ao usuário, apresenta limitações quanto à incapacidade de detectar símbolos apresentando rotação de coordenadas em relação a seus equivalentes no dicionário de matrizes, quanto à limitação de tamanhos e quanto ao longo tempo de resposta e baixa quantidade de símbolos reconhecíveis.

2.5.4.3-Exibição de Informação

E a Exibição de Informação que orienta o usuário para a operação do programa. Através dela, tanto objetos quanto opções de ações são apresentados ao usuário.

A informação deve ser apresentada de maneira efetiva, para promover uma eficiente interação entre o usuário e o computador. Dois tópicos principais devem ser observados :

- LAYOUT GLOBAL : diz respeito à utilização da área da tela. Quando ela não for suficiente para apresentar uma figura completa, técnicas de partição e de seleção das partes devem ser utilizadas. Os menus, "prompts" (avisos ao usuário de que o sistema está disponível para receber comandos ou dados) e outros objetos de controle devem ser o mais compactos possível e devem ser deletados quando não forem absolutamente necessários. Mensagens de erro não necessitam de alocação permanente de espaço e menus de comando só são necessários quando o usuário necessita efetuar seleção de comandos. Mensagens de "prompting" podem ser encurtadas ou suprimidas para usuários experientes. Um expediente interessante é efetuar a divisão da tela em três partes ("windows", ou janelas), por exemplo, uma para "prompts" e erro, outra para menus e outra para a aplicação e deixar ao usuário o controle do tamanho de cada área. Os usuários novatos deixam então mais espaço para informações para operar o sistema ("prompts"), enquanto os experientes, para a aplicação;

- EXIBIÇÃO DOS OBJETOS : diz respeito à seleção da representação para cada tipo de objeto mostrado na tela (tanto intrínsecos quanto de controle), considerando todos

os seus atributos relevantes e as relações entre os objetos. É importante respeitar as convenções gráficas às quais o usuário está acostumado, a economia de espaço na tela e de complexidade na imagem, e evitar erros gramaticais. Alguns trabalhos realizados na área apontam seis componentes da representação gráfica, ou seja, seis variáveis visuais : tamanho, aspecto, orientação, cor, tonalidade e textura, que devem ser analisados no projeto.

Em resumo, para a exibição de informação são necessários : simplicidade, uso econômico do espaço da tela, consistência na representação de objetos e atributos, evitar erros gramaticais e dar atenção ao layout global.

2.5.4.4-Realimentação ao Usuário

A Realimentação permite ao computador informar ao usuário o andamento do trabalho sendo realizado. A realimentação apresenta-se sob a forma de reconhecimento da recepção de comandos, mensagens explanatórias, indicação de objetos selecionados e eco de caracteres digitados e surge como uma resposta às ansiedades do usuário que tem de aguardar o processamento do comando fornecido ao sistema e o consequente resultado. Sem a realimentação, o usuário pode ficar intranquilo e duvidoso quanto à recepção do comando, quanto à inexistência de erros, quanto ao tempo que deverá esperar, etc. Podem se distinguir três tipos de realimentação, todos com restrições de velocidade :

- REALIMENTAÇAO DE COMANDO : Informação sobre a recepção do comando, estágio de execução e condições de erro. Justifica-se da seguinte forma :

  - é possível mostrar o efeito de uma ação do usuário antes que ela seja definitivamente realizada;

  - condições de erro devem ser acusadas imediatamente, através de mensagens breves e claras em locais visíveis;

  - para comandos de execução lenta, deve ser permitido ao usuário saber o tempo necessário ou acompanhar o progresso da execução, o que serve também para indicar que o sistema não está pronto para receber novos comandos.

Para servir de "prompt" a usuários novatos, uma mensagem pode ser mais útil enquanto para usuários experientes, a simples realimentação de cursor (isto é, um cursor representado por um pequeno retângulo ou uma cruz, ao aparecer na tela, ou ao piscar, indica ao usuário a disponibilidade do sistema) pode ser mais útil por ser mais

visível e não exigir deslocamento da vista;

-REALIMENTAÇÃO DE SELEÇÃO : apesar de frequentemente apresentar dificuldade na sua implementação, a realimentação de seleção deve ser provida após a identificação do objeto, através de alguma variação em sua representação no vídeo, que confirme ao usuário a seleção efetuada. Se o usuário move o dispositivo de entrada de coordenadas, a realimentação deve ser removida, permitindo a seleção de novo objeto. Usa-se muito como realimentação de seleção, a mudança de brilho do símbolo, para terminais vetoriais e a inversão da intensidade dos pixels no retângulo envolvente do símbolo, para terminais raster. Assim, a estrutura de dados deve ser projetada de forma a prover rápido acesso aos atributos necessários à realimentação. Isto normalmente é feito definindo-se o conjunto de primitivas que compõe a representação gráfica do símbolo como um segmento único no DISPLAY-FILE, o que permite tratar o símbolo todo como uma unidade, possibilitando modificação de atributos (visibilidade, cintilamento, intensidade e outros relacionados à aparência física) de todo o conjunto de uma vez;

- REALIMENTAÇÃO INDEPENDENTE DA APLICAÇÃO : dois tipos merecem destaque. São eles :

- realimentação de cursor seguindo o dispositivo de entrada; através de software adequado, o cursor é deslocado na tela acompanhando a ação de algum dispositivo de posicionamento ou apontamento, para indicar ao usuário a posição ou o objeto que será devolvido à aplicação como resultado de sua escolha; realimentação deste tipo deve ser rápida pois qualquer atraso é perceptível;

- eco de caracteres digitados; neste caso, a velocidade é de menor importância.

3-SISTEMAS PARA TRATAMENTO DE DIÁLOGO.

Conforme discutido no capítulo 2, a separação entre a aplicação e o diálogo que a controla nos sistemas interativos de computação, com a comunicação usuário-sistema sendo realizada por uma camada independente do programa de aplicação e do pacote gráfico de suporte, traz diversos benefícios, entre os quais (/GIIT 83/) :

- fomenta um trabalho interdisciplinar, com especialistas de aplicação, de núcleo gráfico, de interação, cada um realizando uma parte do projeto do software;

- permite exploração de várias características de dispositivos físicos e de técnicas de interação para atender aos fatores humanos do operador;

- traz economia de custos, facilitando a construção da interface de usuário, permitindo o seu aprimoramento para casos específicos e a sua reutilização por outras aplicações;

- permite consistência através de várias aplicações que podem fazer uso da mesma interface.

A ação interdisciplinar para a produção de interfaces usuário-computador, envolvendo cientistas de computação, psicólogos industriais e cognitivos, projetistas gráficos, artistas e usuários, tem sido apontada como a forma de se obter uma visão de aplicações interativas mais centrada no usuário (/GIIT 83/).

A definição de uma camada de diálogo responsável pelo controle da interação e distinta da camada responsável pela realização das tarefas da área de aplicação do sistema interativo, traz a possibilidade de aproveitar o esforço dispendido na confecção de um diálogo específico para a confecção de diálogos para outras aplicações. Como exposto no item 2.5.1, o programador de aplicação pode projetar suas rotinas considerando que rotinas de interação previamente implementadas estão ao seu dispor para realizar a troca de informações com o usuário.

Por outro lado, nenhum sistema de diálogo pode ser implementado, a priori, suficientemente otimizado para o tipo de aplicação em questão. Assim, surge uma necessidade de avaliação de desempenho da interação operador-sistema e de uma adaptação da estrutura do diálogo às características específicas de um determinado operador e de uma determinada aplicação.

Para suprir esta necessidade de avaliação e adaptação de diálogos, deve-se obter recursos para a definição de diversas estruturas para um mesmo diálogo, para a alteração de passos de

uma determinada estrutura, para controle da execução do diálogo, para coleta de dados sobre o desempenho, para a avaliação destes dados e finalmente para a substituição de uma estrutura de diálogo por outra. Repetindo-se o processo de supervisão de execução e de coleta e avaliação de dados, pode-se determinar qual a estrutura mais eficiente, obtendo-se então uma otimização do trabalho a ser executado pelo operador quando utilizando o sistema.

Todas estas tarefas são referidas coletivamente como "prototyping", entendido como sendo a simulação da execução de um diálogo, podendo acarretar em monitoramento, análise e adaptação do mesmo a uma situação específica e na posterior geração de um produto final, constando da aplicação e seu diálogo otimizado.

Dentro desta filosofia de tratamento de diálogos, têm surgido diversos sistemas que apoiam a definição, o monitoramento e a análise de diálogos (/HANU 82/, /ENCA 83/, /BLAC 77/, /CUNH 81/, /KAIS 82/, /BORU 80/). A maioria destes sistemas, entretanto, foi concebida tendo em vista a sua implementação, para atender a necessidades específicas, sem a preocupação de uma formalização conceitual, dentro da área de pesquisa.

Após o primeiro Workshop sobre técnicas de interação (/GIIT 81/), tais sistemas passaram a ser enfocados sob um ponto de vista mais conceitual, fugindo a considerações de implementação. Tal aproximação permite concentrar-se nos fatores humanos, ou seja, preocupar-se com o usuário e com sua capacidade de trabalho, muitas vezes subestimada pelos programas interativos. Desta aproximação surge o conceito de UIMS (USER INTERFACE MANAGEMENT SYSTEM), sistema de gerenciamento de interfaces de usuário (/GIIT 81/).

Neste trabalho foi adotada uma posição equidistante em relação a estas duas visões, a dos sistemas implementados e a de UIMS. É proposta uma ferramenta, o PROCESSADOR DE DIÁLOGO, que pretende auxiliar o projetista na produção de diálogos, permitindo definição, "prototyping" e geração de um diálogo otimizado. A ferramenta possibilita ao projetista executar um projeto estruturado e, desde que este projetista tenha em mente as considerações de fatores humanos, ele está capacitado a traduzir em seu diálogo um sistema fácil e agradável ao usuário. Juntamente com uma formalização teórica do Processador, são apresentadas questões relacionadas com sua implementação, que se apresentam como trabalhos para futuro desenvolvimento.

Justamente por sua capacidade de apoio ao projeto de sistemas de diálogo, pode-se citar a área de Aplicação de Metodologia de Projeto de Software como sendo a área de aplicação do Processador de Diálogo. Pretende-se, com sua definição, contribuir para a obtenção de diálogos, para sistemas interativos, que sejam independentes de aplicação, da realização física dos dispositivos de entrada e de saída e das funções gráficas de suporte. O Processador de Diálogo é uma ferramenta que permite selecionar a melhor estratégia para operadores

específicos operarem programas específicos. Obtendo-se otimização da interação operador-programa, o Processador de Diálogo configura-se como um sistema gerador de protótipos (PROTOTYPING), dada sua capacidade de auxiliar na avaliação de diversas possibilidades para a estrutura de um determinado diálogo e na opção pela de major efetividade. Uma vez identificada a estrutura ótima para o diálogo entre um operador e uma aplicação específicos, o Processador de Diálogo está apto a gerar seu produto final, qual seja um sistema de aplicação envolvido com seu diálogo otimizado, onde todos aqueles componentes do Processador, utilizados no "prototyping" mas não mais necessários à execução do sistema interativo são subtraídos.

Neste capítulo, é descrito um Processador de Diálogo baseado em idéias expostas na literatura, acrescido de funções no que diz respeito à supervisão de execução, à otimização da interação e à linguagem de especificação de estruturas de diálogo utilizada.

Pode-se visualizar três fases na utilização do Processador de Diálogo : definição, "prototyping" e geração.

Na fase de definição, um diálogo específico é definido. Para a sua representação é utilizado o método de Células de Diálogo, em razão de sua maior potencialidade de representação (que pode ser facilmente assimilada da comparação entre os diversos métodos apresentados no Item 2.2, em razão de seus inúmeros recursos e de sua concepção específica para a finalidade de representação de diálogos). Para a utilização por parte do autor de diálogos, as Células de Diálogo são envolvidas em uma linguagem de programação que pode ser independente ou construída sobre uma linguagem já existente. A linguagem deve manter todas as características das células de diálogo, de forma a permitir a definição de qualquer tipo de diálogo, independentemente do tipo e da complexidade da aplicação à qual este diálogo se destina. Ainda na fase de definição, serão obtidos os programas objeto correspondentes às Celulas de Diálogo e armazenados em uma Biblioteca de Diálogos para uso na fase de "prototyping". Podem ser armazenados diversos conjuntos de descrições para uma mesma aplicação, com graus de dificuldade diferentes. As descrições são analisadas e alterações podem ser feitas em qualquer das células de diálogo.

Na fase de "prototyping", o Processador supervisiona o desenrolar da interação usuário-sistema podendo efetuar coleta e monitoramento de dados e posterior análise destes dados (estatística e de desempenho).

A partir das análises dos dados coletados, a adaptação do diálogo ao usuário pode ser realizada :

- através de alterações na estrutura do diálogo que o tornem mais simples ou mais inteligente (adaptação "off-line", já que, para tal deve-se retornar à fase de Definição de Diálogo) ;

- ou através da escolha dentre dois ou mais conjuntos de descrições (armazenados e catalogados na Biblioteca de Diálogos) daquele que melhor se adapte ao usuário (adaptação "on-line", já que o próprio Processador em execução pode se incumbir da substituição dos conjuntos de descrições).

Na fase de Geração, é obtido o sistema executável otimizado, e colocado à disposição do usuário para realizar a tarefa de aplicação.

Cada uma destas fases é detalhada no item 3.1, com relação às funções envolvidas, e no item 3.2, com relação a cada componente necessário. A visão em 3.1 é de definição funcional e em 3.2, de definição topológica. O item 3.2 ainda define a Linguagem de Especificação de Diálogos (item 3.2.1) e o item 3.3 fecha o capítulo apresentando as conseqüências surgidas da utilização de núcleos gráficos padrões por Sistemas para Tratamento de Diálogos.

3.1-DEFINIÇAO FUNCIONAL DE UM PROCESSADOR DE DIÁLOGO

O Processador de Diálogo é uma ferramenta colocada à disposição de autores de diálogo, especialistas em interação, que o utilizarão no apoio ao projeto de um diálogo a ser utilizado para uma aplicação específica.

O Processador de Diálogo deve então oferecer ao autor de diálogos recursos utilizáveis em cada uma das três fases de projeto de um diálogo, quais sejam, a definição, o "prototyping" e a geração do diálogo otimizado.

3.1.1-Definição de Diálogos

Neste grupo encontram-se as funções relacionadas à criação da estrutura do diálogo, quais sejam, o MODELAMENTO DO DIALOGO, o PRE-PROCESSAMENTO e a ANALISE DA DEFINIÇAO.

3.1.1.1-Modelamento do Diálogo

Para permitir a realização do Modelamento do Diálogo, o Processador de Diálogo deve oferecer uma Linguagem de Especificação de Diálogos (LED) que contenha como primitivas elementos para descrição, em alto nível, de todos os passos de um determinado diálogo.

A linguagem deve ter facilidades para a especificação de todas as características de um diálogo, tanto do ponto de vista das estruturas de controle como das estruturas de informação necessárias ao desenrolar do diálogo. Destacam-se como necessidades a serem satisfeitas :

- descrição sintática das variáveis e dados a serem utilizados;

- construções de controle poderosas, de preferência estruturadas (como por exemplo, DO WHILE, DO UNTIL, SELECT, etc) que permitam a especificação do controle do fluxo do diálogo;

- especificação das opções permitidas ao usuário e das respostas a serem devolvidas a ele;

- especificação do layout completo da tela :

  - como dividir a tela em áreas para trabalho e para comunicação entre o usuário e o sistema (prompts, menus, ecos, etc) ;

- quais objetos (figuras e/ou textos) deverão ser exibidos em cada uma destas áreas.

- manipulação total dos objetos criados e de suas características. Possibilidades de :

  - fornecer parâmetros para a identificação dos objetos criados ;

  - especificar os atributos destes objetos (visibilidade, cintilamento, etc) ;

  - modificar os atributos de objetos já existentes;

  - deletar objetos ;

  - definir tempo de vida dos objetos (duração dos objetos na tela) ;

  - criar cópias idênticas ou modificadas de objetos já existentes.

A necessidade destas facilidades surge para permitir a definição não só da interface entre aplicação e diálogo (ou seja, como, quando e através de que, as funções de aplicação acionam a camada de diálogo para solicitar interação com o operador e a camada de diálogo devolve estes dados à aplicação) como também da interface entre operador e sistema (ou seja, através de que elementos se realiza a comunicação entre operador e sistema).

3.1.1.2-Pré-Processamento e Análise

Como a linguagem para a especificação de diálogos, descrita no Item 3.2.1, é definida como uma extensão a uma linguagem de alto nível, é necessária a existência de um Pré-Processador para adequar o código original à linguagem base e então utilizar o compilador base para a obtenção das descrições de diálogo em sua forma objeto. Durante o pré-processamento, é efetuada uma análise do código para detecção de possíveis erros de sintaxe, os quais podem ser corrigidos através do retorno à função de modelamento.

3.1.2-"Prototyping"

O "Prototyping" envolve a execução e avaliação de protótipos de diálogo. Através do "prototyping" uma aplicação é realizada e toda a comunicação necessária com o operador é

executada de diferentes formas, segundo diferentes fluxos, o que permite avaliar as diversas possibilidades de implementação de cada interação entre a aplicação e o operador.

É necessária, então, a existência de funções de supervisão que permitam ao projetista executar diversas estruturas de diálogo, coletar quaisquer tipos de dados, analisar os dados coletados, armazenar dados de análise, e assim por diante, de tal forma que o próprio projetista de diálogo possa definir funções de apoio ao "prototyping" (além das que eventualmete sejam implementadas pelo implementador do Processador de Diálogo).

São exemplos de funções de interesse :

- executar um determinado utilitário de apoio ao "prototyping";

- executar uma determinada sequência de diálogo para atender a uma solicitação de interação efetuada pela aplicação;

- desativar uma determinada sequência de diálogo (ou seja, um determinado conjunto de descrições) e executar outra (ou seja, outro conjunto de descrições) em substituição ao inicial, ambos definidos para atender à mesma solicitação da aplicação mas com graus de complexidade diferentes.

- obter informações sobre o desenrolar do diálogo com vistas à análise de desempenho ;

- obter informações sobre a capacitação do operador ;

- obter informações sobre o ambiente de trabalho ;

- criar um arquivo que documente os passos dados pelo operador para manter um histórico da sessão de trabalho ;

- quantificar o tempo de resposta do usuário a um pedido de comandos ou dados ;

- definir uma forma de monitoramento ao próprio operador para cientificá-lo do andamento do trabalho.

- analisar o desempenho operador/diálogo, a partir de dados coletados;

- definir formas do Processador de Diálogos interferir no curso do diálogo, notificando ao operador perspectivas de ocorrência de situações indesejadas, de forma a torná-lo mais cuidadoso na operação do sistema ;

- da mesma forma que no item anterior, definir procedimentos de emergência;

- criar mecanismos para tratamento de erros ;

- definir formas de recuperação para o operador, de escolhas erradas ou precipitadas ;

- interromper o diálogo em curso, caso ele esteja caminhando em direções indesejadas.

Além de análises on-line em relação à execução dos protótipos de diálogo, pode existir uma ANÁLISE DE RESULTADOS, que é a função que permite uma análise global e final, realizada "off-line" em relação à sessão de trabalho, com o objetivo de avaliar todo o processo de diálogo.

Uma análise visual por parte do autor de diálogos, com base nos dados fornecidos e análises efetuadas por todas as ferramentas de "prototyping" acionadas quando da execução do diálogo, pode ser a forma mais objetiva de avaliação.

É possível, porém, que diversos pontos sejam analisados automaticamente por um utilitário do Processador de Diálogo, após cada sessão de trabalho, possibilitando detectar então necessidades de alterações nas descrições de diálogo.

Uma adaptação "off-line" pode ser realizada, através de alterações em descrições de diálogo ou através de modificação no conjunto de descrições a serem ligadas quando da próxima etapa de obtenção de protótipos de diálogo.

Em resumo, são gerados diversos protótipos de diálogo para uma aplicação. Estes protótipos sao executados, avaliados e adaptados, e aquele cujo desempenho mais se aproximar da expectativa do autor de diálogos é adotado como produto final.

3.1.3-Geração do Diálogo Otimizado

Na fase de GERAÇÃO, são liberados do sistema executável todos aqueles componentes do Processador de Diálogo cuja atuação só se justifica na realização do "prototyping". Além disso, são eliminadas das Células de Diálogo fonte as primitivas específicas de "prototyping".

É gerado então um novo sistema, através da ligação de um Supervisor de Execução, do programa de aplicaçao, de rotinas do núcleo gráfico e de seqüências de diálogo, que se presta a ser executado e permitir ao operador a realização das tarefas da área de aplicação.

3.2-IMPLEMENTAÇÃO DE UM PROCESSADOR DE DIÁLOGO

Neste item é descrita a topologia adequada ao Processador de Diálogo que pretenda realizar todas as funções apresentadas no item 3.1. Ao contrário daquela visão funcional, o enfoque dado aqui é o de implementação e o objetivo é levantar alguns dos principais pontos a serem considerados num futuro trabalho de implementação do sistema Processador de Diálogo.

Dos componentes do Processador de Diálogo, alguns realizam funções off-line em relação à execução do protótipo de diálogo (antes ou depois dela) e outros funções on-line. Assim, são apresentadas topologias adequadas para cada uma das fases de utilização do Processador, definição, "prototyping" e geração, apresentadas na introdução deste capítulo.

Na primeira fase, é realizada a definição de diálogos. Como resultado da escolha do método de representação por Células de Diálogo, cada passo em cada interação fica a cargo de uma Célula de Diálogo, que na prática, corresponde a uma rotina. Assim, a estrutura do diálogo está descrita através de diversas rotinas, as quais devem ser submetidas ao computador. Um editor de texto é utilizado para submeter as Células de Diálogo ao computador num formato tratável por ele. Mas, estas Células de Diálogo estão descritas pela linguagem fonte e por ser ela uma linguagem especial, uma extensão do FORTRAN-77 (como é descrito no item 3.2.1), não são providos compiladores que as codifiquem em forma objeto.

Assim, deve-se utilizar um pré-processador que leia as Células de Diálogo, interprete-as e gere rotinas em FORTRAN-77, através do mapeamento das primitivas extendidas da LED às primitivas básicas do FORTRAN-77. Após este pré-processamento, as rotinas podem ser submetidas ao compilador FORTRAN-77 que se encarrega de gerar rotinas em linguagem objeto, ou seja, as próprias Células de Diálogo objeto, que são ligadas para obtenção do programa executável. Faz parte da função do pré-processador a análise do código que permita a detecção de erros de digitação quando da criação dos arquivos e de falhas na observação das regras da linguagem.

Cada conjunto de células de diálogo capaz de realizar uma interação com o operador é reunido sob a forma de um subprograma (subprograma diálogo). Assim, podem-se ter diversos subprogramas diálogo capazes de atender a cada pedido de dado realizado pela aplicação. A otimização do diálogo está em se determinar, para cada interação, qual é o subprograma mais adequado.

Além das Células de Diálogo, são necessários à execução do diálogo, o programa de aplicação que fará uso do diálogo (ou um simulador dele, caso se pretenda testar o diálogo separadamente ou previamente em relação ao desenvolvimento do programa de aplicação), as rotinas do núcleo gráfico, que dão

apoio à geração e manipulação de figuras para a comunicação com o operador, o Supervisor de Execução que controla a execução do "prototyping" e rotinas (na forma de FUNCTION's) de apoio ao "prototyping" (para coleta de dados, análise, etc, algumas providas pelo implementador do Processador de Diálogo e outras pelo próprio projetista de interação). Por hipótese, todos estes elementos já estão disponíveis em sua forma objeto.

Surge a necessidade de se obter a tarefa aplicação, ou seja, o programa de aplicação ligado ao supervisor de execução, aos subprogramas de diálogo, às rotinas do núcleo gráfico e aos utilitários de "prototyping". Esta necessidade é suprida pelo próprio ligador do sistema operacional do computador hospedeiro.

A figura 3.1 apresenta a sequência de atividades desde a criação das células de diálogo até a obtenção da tarefa aplicação.

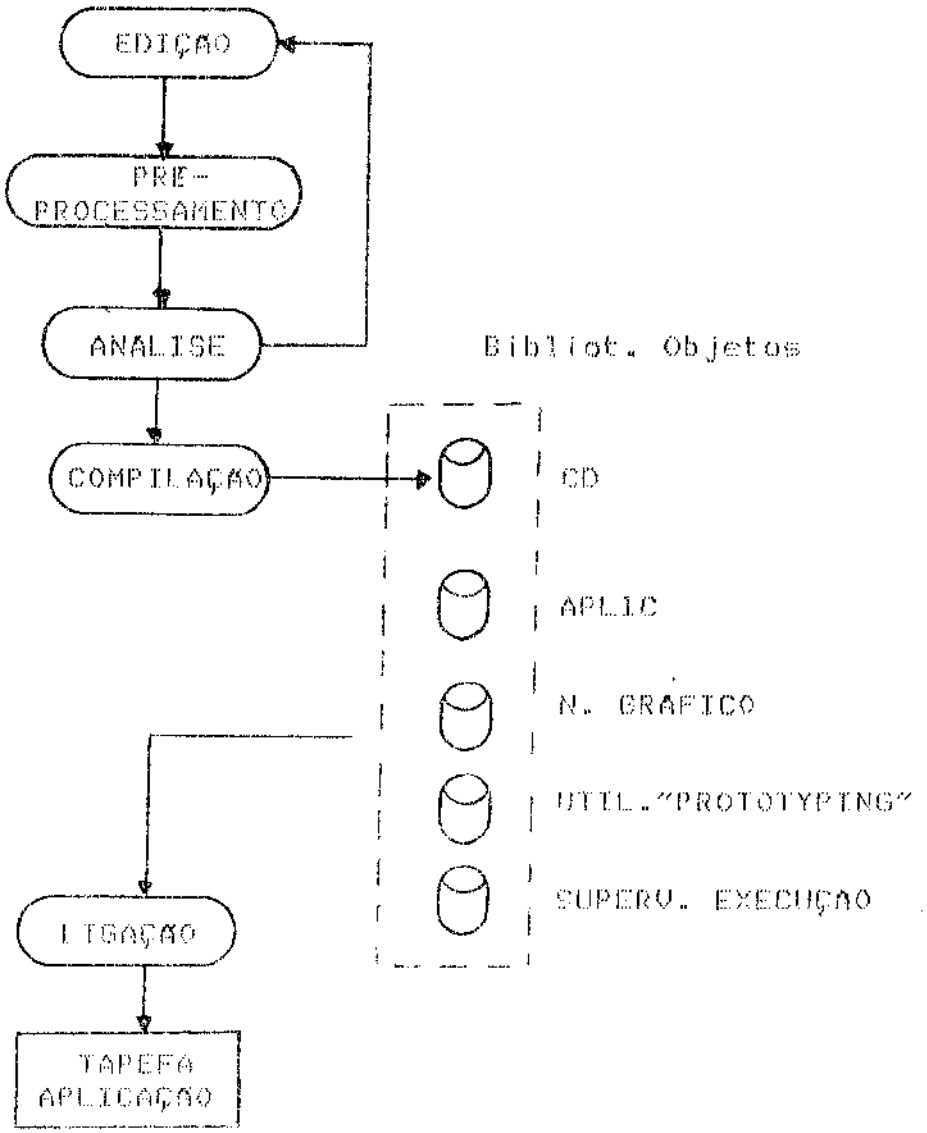


fig. 3.1 : Fase de definição.

Na segunda fase, já obtida a tarefa aplicação, deve ser realizado o "prototyping". A tarefa aplicação é executada e o supervisor de execução é acionado toda vez que a aplicação desejar se comunicar com o usuário. O supervisor se configura então como a ligação entre a aplicação e o diálogo. O supervisor se incumbe de acionar um dos subprogramas diálogo providos para aquela interação e passar-lhe o controle de execução. O subprograma é executado e, nos locais definidos pelo projetista de interação, são ativadas funções de "prototyping" (providas na forma de FUNCTION's) que realizarão coleta e análise de dados (e outras atividades previstas pelo projetista de interação). Quando o controle é devolvido ao supervisor de execução ele aciona outro subprograma diálogo para a mesma interação e repete-se o processo que permite então a avaliação dos subprogramas diálogo projetados para aquela interação. A seguir, o controle é devolvido ao programa de aplicação que retoma sua execução. A cada nova interação, ou seja, a cada nova necessidade de comunicação entre o programa de aplicação e o operador, um novo conjunto de subprogramas diálogo é executado, um subprograma por vez, permitindo avaliar os diversos subprogramas providos para cada interação. Ao final da execução da tarefa aplicação, pode-se determinar, pelos dados coletados, qual subprograma é apropriado para cada interação com a aplicação em questão e o conjunto destes subprogramas constitui o diálogo otimizado para aquela aplicação.

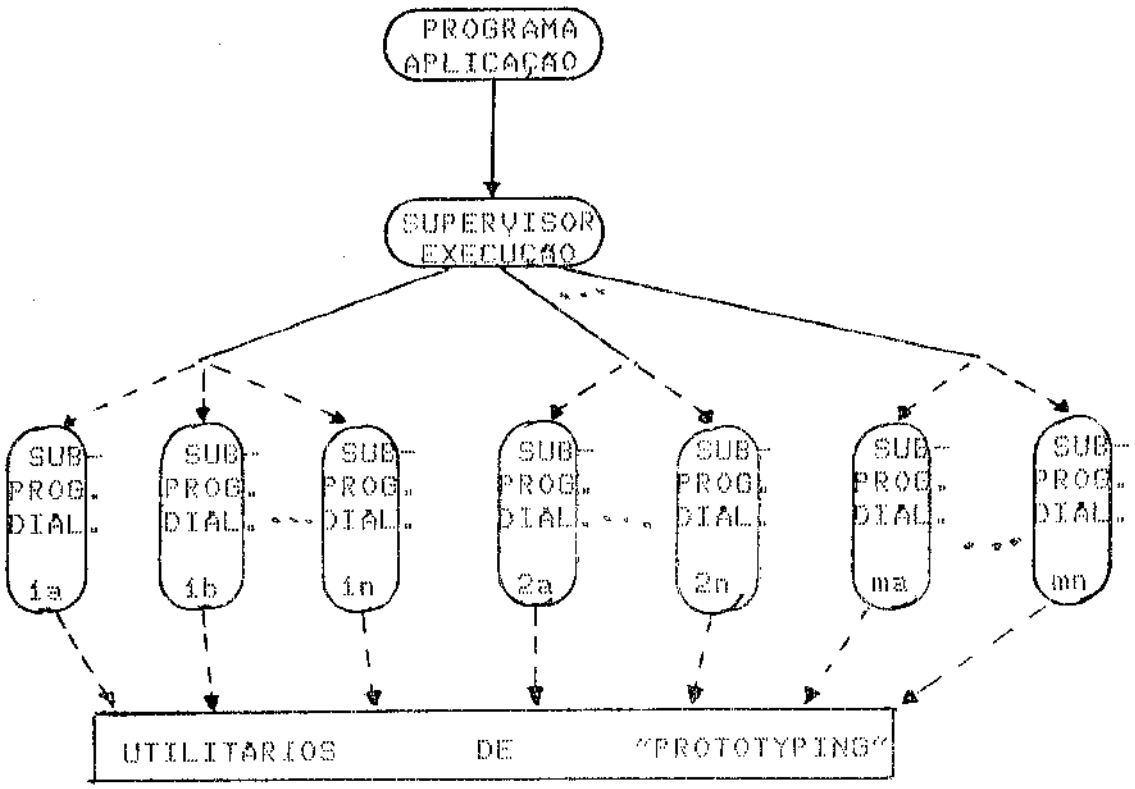


fig. 3.2 : Fase de "prototyping"

A figura 3.2 apresenta o diagrama da tarefa aplicação sendo executada, onde são exemplificadas m interações com n possibilidades de sequências de diálogo diferentes para cada uma das interações.

Na terceira fase, de geração, um elemento, dito LIMPADOR, é responsável por processar os subprogramas diálogo selecionados na fase anterior e eliminar das células de diálogo correspondentes todas as primitivas específicas de "prototyping", desnecessárias à operação normal do diálogo.

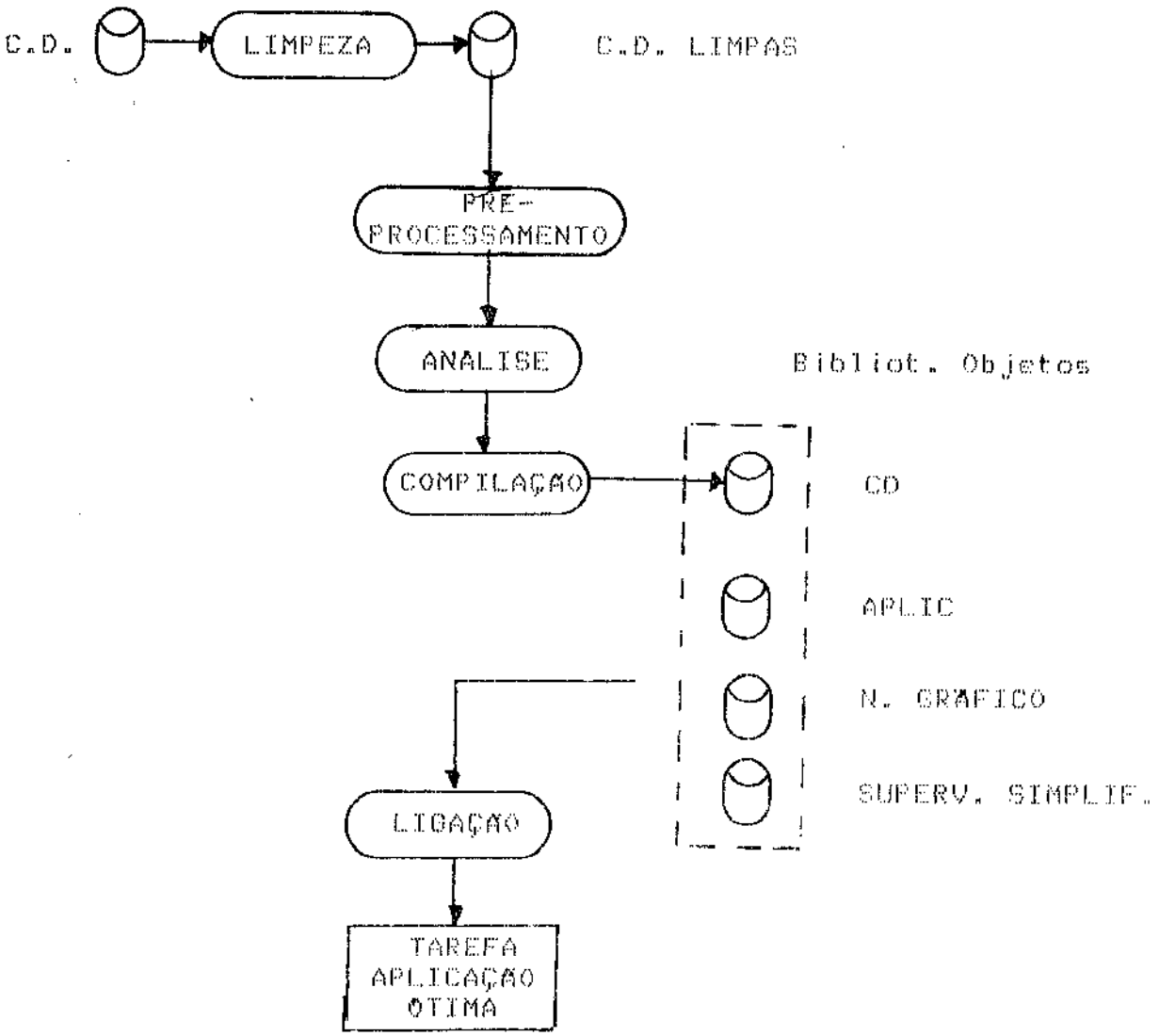


fig. 3.3 : Fase de geração.

As células de diálogo resultantes são novamente pré-processadas, compiladas e ligadas, agora ao programa de aplicação, ao supervisor de execução e às rotinas do núcleo gráfico. Deve-se notar que o supervisor de execução agora utilizado é uma versão simplificada daquele utilizado para o "prototyping", já que agora não são mais necessários os testes e controles para acionar diversas sequências de diálogo para atender a cada solicitação de interação efetuada pela aplicação. Já foi identificada a sequência ótima para cada interação e o supervisor simplesmente faz a interface entre a aplicação e as sequências ótimas. A operação de obtenção da tarefa aplicação ótima é ilustrada na figura 3.3.

Na sequência, são detalhadas a Linguagem de Especificaçao de Diálogos, no item 3.2.1, o pré-processador, no item 3.2.2, e o supervisor de execução, no item 3.3.3. O item 3.2.4 apresenta exemplos de utilitários de "prototyping", como Coletor de Dados, Monitor e Analisador e o item 3.2.5 descreve o Limpador.

3.2.1-A LINGUAGEM DE ESPECIFICAÇÃO DE DIÁLOGOS (LED)

Deve ser definida uma linguagem que permita ao autor de diálogo a definição da estrutura de um diálogo. Esta linguagem refletirá uma das formas de representação expostas no item 2.2. A partir da decisão de se adotar o método de Células de Diálogo, por ser, entre os métodos apresentados no item 2.2, o mais completo e específico para descrição de diálogos, a LED se baseará nele. Assim, todas as características expostas no item 2.2.3 são agora incorporadas à LED que deverá então permitir que cada passo de um diálogo seja descrito através de uma Célula de Diálogo cuja forma obedecerá àquela apresentada em 2.2.3.

Deve-se notar que o método de Células de Diálogo não se configura, por si só, como uma linguagem computacional de especificação de diálogos pois, apesar de apresentar uma sintaxe, através de regras de produção, faltam-lhe estruturas para a especificação e controle do fluxo dos elementos de diálogo sendo definidos.

A linguagem LED consiste de blocos, onde o mais externo é a Célula de Diálogo que é a unidade para compilação. Células não podem conter outras células mas podem ativá-las, tornando-as suas subcélulas. Dentro de cada célula, blocos podem conter blocos, desde que sem superposição parcial. Os blocos podem ser precedidos por condições, podem estar dentro de loops ou podem conter loops e construções condicionais. Saltos são permitidos desde que não para dentro ou para fora de um bloco. Como se vê, a linguagem objetiva permitir a melhor estruturação possível das declarações.

São preocupações na definição da LED:

- facilidade de aprendizado por parte dos autores de diálogo;
- existência de regras simples e consistentes;
- liberdade na composição das declarações.

E com relação à implementação :

- independência de máquinas (computadores);
- independência de dispositivos (terminais);
- portatilidade.

Para se ter a portatilidade do Processador de Diálogo em relação a computadores, a linguagem escolhida deve ser uma linguagem largamente conhecida e utilizada. A experiência tem mostrado que a maioria das linguagens padrão de hoje em dia não é apropriada para a descrição de estruturas de diálogos e que não é vantajosa a definição ou criação de uma linguagem específica para

diálogos. A melhor solução seria, portanto, a extensão de uma linguagem padrão que permitisse o uso de seu compilador após a passagem por um pré-processador cuja construção é relativamente simples. Este pré-processador poderia se incumbir não só de substituir construções lógicas avancadas por outras existentes na linguagem padrão, como da conversão de tipos de dados específicos de diálogo (POINT, VIEWPORT, PICTURE, etc) para os tipos existentes na linguagem padrão e também da interpretação das palavras reservadas utilizadas na definição do diálogo (exemplo : INIT END, DDECL END, BODY, etc.). As construções de controle da LED foram definidas a partir da linguagem FLECS (assim como feito em /BORU 81/), em razão de sua clareza e objetividade. Para linguagem base, foi escolhida a linguagem FORTRAN-77, devido a sua disponibilidade, ao emprego relativamente amplo e às suas construções estruturadas.

Os recursos oferecidos pela LED podem ser divididos em três grupos : ESTRUTURAÇÃO DO DIALOGO, SUPERVISÃO DE EXECUÇÃO, COMUNICAÇÃO, que são detalhados a seguir.

3.2.1.1-Estruturação do Diálogo

A estruturação do diálogo é realizada através de quatro tipos de primitivas, descritas abaixo.

- primitivas para declaração de dados : permitem a declaração das variáveis a serem utilizadas numa determinada Célula de Diálogo assim como seu tipo de dado e valores iniciais. Todos os elementos do diálogo são definidos através destas primitivas. Dividem-se em quatro grupos :

  - declaração de parâmetros (transmitidos na chamada das células) :

    - IN : de entrada;
    - OUT : de saída;
    - TRANS : transientes.

  - declaração de dados (acesso por declaração) :

    - LOC : locais;
    - GLOB : globais.

  - definição do tipo de dado :

    - INTEGER : dados inteiros;
    - REAL : dados reais;
    - BOOLEAN : dados lógicos;
    - STRING : cadeias de caracteres;
    - POINT : coordenadas de um ponto;
    - PICTURE : identificação de uma figura;
    - PROMPT : identificação de um objeto de prompt;
    - ECHO : identificação de um objeto de eco;
    - VIEWPORT : reais identificando uma janela de exibição.

- atribuição de valores iniciais a variáveis do tipo INTEGER, REAL, BOOLEAN, STRING ou POINT :

- INIT ... INIT END.

- primitivas de controle da estrutura : dão forma às células de diálogo, orientando a estrutura das declarações. Dividem-se em seis grupos :

- DIACELL : indica o início de uma célula de diálogo. É seguida pelo nome da célula;

- IN, TRANS, OUT, PDECL END : indicam o bloco de declaração dos parâmetros da célula;

- LOC, GLOB, DDECL END : indicam o bloco de declaração dos dados utilizados na célula;

- INIT ... INIT END : indica o bloco de inicialização de variáveis;

- BODY : especifica o corpo da célula, onde se localiza a sua função basica;

- END DIACELL : indica o fim de uma célula de diálogo.

- primitiva de sequenciamento de eventos : permite a ativação e desativação de subcélulas de diálogo, a partir da célula correntemente em execução. Atua como um mecanismo de hierarquização de células, pois permite a ativação, a partir de uma célula, de subcélulas, com consequente passagem de parâmetros (é o conceito de PROCEDURE):

- SYMBOL

- primitivas de estruturação dos algoritimos: permitem a utilização de blocos de comandos com ligações de hierarquia entre si, assim como a seleção de caminhos dentro da célula, a partir de testes e condições. Permitem assim, o controle do fluxo de dados dentro de cada célula. Para a apresentação das primitivas, serão descritas as construções na LED, as construções equivalentes adotadas em FORTRAN-77 e os fluxogramas elucidativos. Serão consideradas as seguintes convenções :

- C : expressão lógica;

- S : escopo de uma ou mais declarações ;

- E : qualquer tipo de expressão (inteira, real, etc.);

- I : especificação de qualquer incremento legal;

- v : variável real ou inteira;

- e1,e2,e3 : expressões aritméticas.

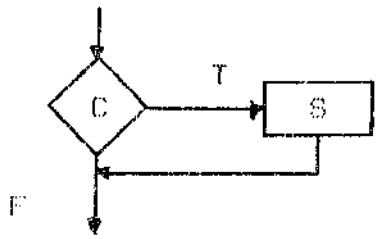
São dois os tipos de primitivas de estruturação de algoritmos :

- primitivas de decisão : São construções que controlam a execução das declarações através de expressoes lógicas ou testes. São elas :

IF : avalia uma expressão lógica que, se for verdadeira, determina a execução da declaração.

```
IF (C) S

  ou

IF (C)
.  S1
.  S2
...FIN
```

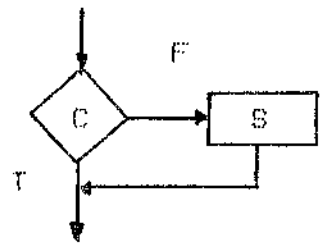
```
IF (C) S

  ou

IF (C) THEN
     S1
     S2
END IF
```



UNLESS : é equivalente a IF(.NOT.C) mas é mais conveniente.

```
UNLESS (C) S

  ou

UNLESS (C)
.  S1
.  S2
...FIN
```

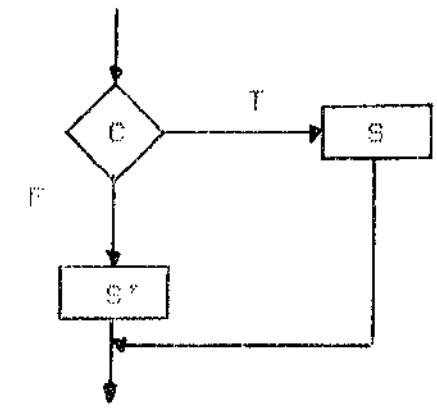
```
IF (.NOT.C) S

  ou

IF (.NOT.C) THEN
     S1
     S2
END IF
```



WHEN/ELSE : se a expressão lógica é verdadeira, as declarações relativas a WHEN são executadas; senão, as declarações relativas a ELSE são executadas.

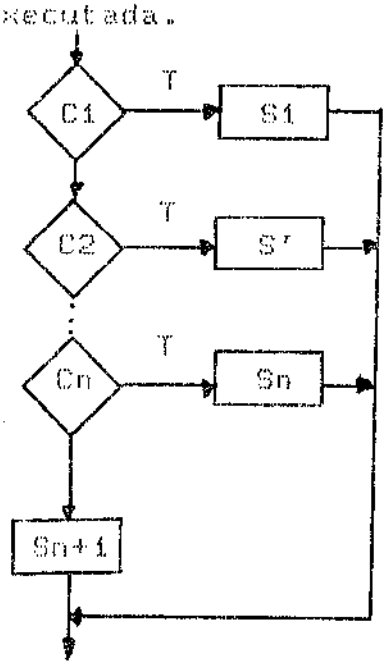
```
WHEN (C) S1
ELSE     S2

  ou

WHEN (C)
.  S1
.  S3
...FIN
ELSE
.  S2
.  S4
...FIN
```

```
IF (C) THEN
     S1
     S3
ELSE
     S2
     S4
END IF
```



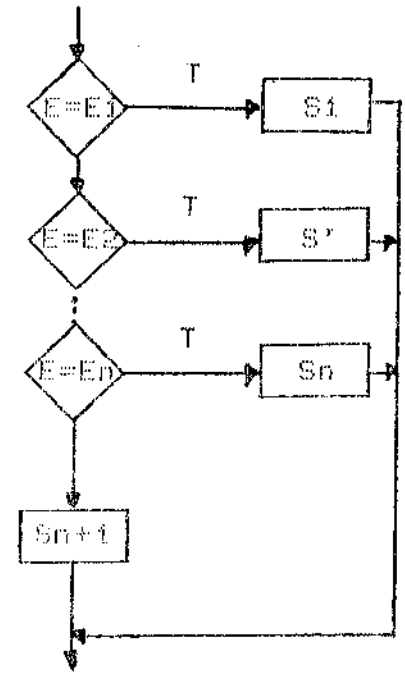
CONDITIONAL : diversas expressões lógicas são avaliadas, uma por uma, até que a primeira verdadeira seja encontrada, executando-se então a declaração correspondente. Se nenhuma das expressões é verdadeira, a declaração correspondente a OTHERWISE é executada.

```
CONDITIONAL                 IF (C1) THEN
.  (C1) S1                      S1
.  (C2)                      ELSEIF (C2) THEN
   .  S2                        S2
   .  S3                        S3
   ...FIN                    ELSEIF (Cn) THEN
.  (Cn) Sn                      Sn
.  (OTHERWISE) Sn+1          ELSE
...FIN                           Sn+1
                            END IF
```



SELECT : é similar ao CONDITIONAL: Uma determinada expressão E é comparada a cada expressão Ei numa lista e somente a declaração correspondente à primeira expressão encontrada que seja igual a E, é executada. A opção pelo OTHERWISE também pode ser feita. Como a expressão E é avaliada sempre que utilizada, se ela é complexa deve-se pré-computa-la, atribuir seu valor a uma variável e utilizar esta variável no comando SELECT.

```
SELECT (E)                  IF (E.EQ.E1) THEN
.  (E1) S1                      S1
.  (E2)                      ELSEIF (E.EQ.E2) THEN
   .  S2                        S2
   .  S3                        S3
   ...FIN                    ELSEIF (E.EQ.En) THEN
.  (En) Sn                      Sn
.  (OTHERWISE) Sn+1          ELSE
...FIN                           Sn+1
                            END IF
```



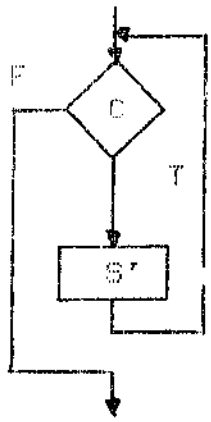
- primitivas de loop : Uma declaração ou conjunto de declarações, é executada uma quantidade variável de vezes que depende de condições especificadas. São elas:

DO : controla a quantidade de vezes que um conjunto de declarações será executado.

```
DO  (I)  S              DO  v=e1, e2 (,e3)

    ou                      S1

DO (I)                      S2
.  S1
.  S2                  END DO
...FIN
```

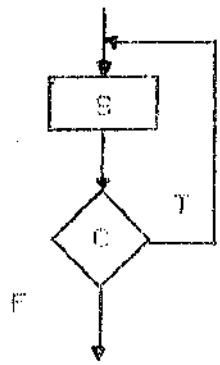
WHILE : repete a execução das declarações "enquanto" uma condição especificada é verdadeira. A condição é avaliada antes da primeira execução, permitindo que nem haja execução se ela for inicialmente falsa.

```
WHILE  (C) S            DO  WHILE (C)

       ou                       S

WHILE  (C)              END DO
   .  S1
   .  S2
   ...FIN
```



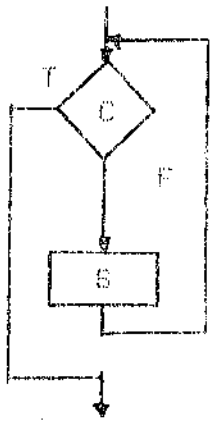
REPEAT WHILE : similar ao WHILE, só que neste caso, a condição só é avaliada após a primeira execução, exigindo que haja sempre ao menos uma execução da declaração, mesmo que a condição seja inicialmente falsa.

```
REPEAT WHILE  (C) S                S
                              DO  WHILE (C)

       ou                          S

REPEAT WHILE  (C)             END DO
   .  S1
   .  S2
   ...FIN
```



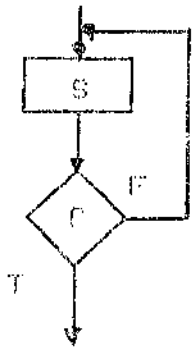
UNTIL : repete a execução das declarações "até" que a condição especificada seja verdadeira. A condição é avaliada antes e então se a condição é inicialmente verdadeira, nem haverá execução da declaração.

```
UNTIL (C) S          DO  WHILE (.NOT.C)
        ou                    S1
UNTIL (C)                     S2
   . S1
   . S2              END DO
   ...FIN
```



REPEAT UNTIL : similar ao UNTIL, com o teste sendo movido ao fim do loop, obrigando a execução da declaração sempre, mesmo que a condição já seja verdadeira.

```
REPEAT UNTIL (C) S          S
                       DO  WHILE (.NOT.C)
        ou                    S
REPEAT UNTIL (C)            END DO
   . S1
   . S2
   ...FIN
```



3.2.1.2-Supervisão de Execução

Para a Supervisão da Execução do diálogo, o autor de diálogo tem à sua disposição uma primitiva que lhe permite definir diversas formas de apoiar o "prototyping" do diálogo. Conforme apresentado no item 3.2, o "prototyping" ocorre com a execução de diversos subprogramas diálogo e de diversos utilitários de "prototyping" (implementados na forma de FUNCTION's), que podem ser providos pelo implementador do Processador de Diálogo, pelo instalador do sistema, ou pelo próprio projetista de interação. Assim, é necessária uma primitiva que permita a execução de FUNCTION's e ela é :

- FUNTION 'NAME' : primitiva de ativação de utilitários de

"prototyping". Permite executar subprogramas de supervisão, como por exemplo, Coleta de Dados, Monitoramento, Analise, Coleta de informações sobre o estado do diálogo, etc.

3.2.1.3-Comunicação

A LED oferece ao autor de diálogo primitivas especificas para a comunicação entre o usuário e o computador. Estas primitivas permitem a definição da forma de comunicação, ou seja, a definição da própria interface de usuário, que especifica com que elementos se fará a troca de informações entre operador e sistema. Existem cinco grupos de primitivas de comunicação. A sintaxe completa de cada primitiva destes cinco grupos está exposta no item 2.2 (através da definição das células de diálogo), à exceção das primitivas de saída, que dependiam da escolha do núcleo gráfico de suporte ao Processador de Diálogo. Assim, em anexo à descrição das primitivas de saída, é apresentada sua sintaxe. Os cinco grupos são :

- definição de elementos de comunicação : a comunicação se faz através de pedidos de dados (prompt) e indicação do recebimento dos dados, ou seja, a realimentação (echo). Cada prompt ou eco é composta de texto, figura ou menu de escolhas (este último só para prompt), deve ser exibida em uma determinada janela na tela e tem atributos vinculados à sua representação gráfica :

  - PROMPT DEF ... DEF END : define uma prompt, através de um nome, uma janela associada, atributos, duração e o objeto a ser utilizado (figura, texto ou menu);

  - ECHO DEF ... DEF END : similar a prompt exceto para o objeto de menu;

  - PICTURE DEF ... DEF END : define uma figura, através de um nome, uma janela, atributos e funções de saída gráfica responsáveis pela geração da representação visual;

  - MENU FOR CHOICE : define um menu formado de strings de texto então definidas ou figuras já definidas como PICTURE;

  - TEXT : define uma string de caracteres a ser utilizada como prompt ou eco ou elemento de menu de escolhas.

- modificação de atributos dos elementos : assim como a simples apresentação de elementos tem para o operador um significado especifico, a alteração de atributos visuais destes elementos também pode ter. Atributos dinâmicos de

prompt e eco são visibilidade, cintilamento e transformação de coordenadas (rotação, translação e escalamento), que podem ser alterados quando necessário :

- PROMPT MOD ... MOD END : permite modificar visibilidade e cintilamento e transformar coordenadas, de acordo com uma matriz de transformação, de objetos de prompts;

- PROMPT MENU MOD ... MOD END : permite modificar visibilidade de menus que representem prompts;

- ECHO MOD ... MOD END : permite alterar visibilidade e cintilamento e transformar coordenadas de qualquer objeto de eco.

- deleção de elementos : prompts e ecos podem ser explicitamente deletados, independente do tempo de vida especificado quando de sua criação :

- PROMPT DELETION : permite deletar uma prompt;

- ECHO DELETION : permite deletar um eco.

- primitivas de saída : estão assentadas sobre primitivas e atributos de primitivas de saída do núcleo gráfico e se responsabilizam pela saída gráfica :

- LINE : permite a geração de linhas conectadas a partir de um conjunto de pontos com atributos de intensidade, tipo e espessura de linha;

- MARKER : permite a geração de marcadores em posições dadas, com atributos de intensidade, tipo e tamanho do marcador;

- TEXT : conforme definida no subitem de definição de elementos de comunicação;

- AREA : permite a geração de uma área definida por um conjunto de pontos com atributos de intensidade e estilo do interior (forma de preenchimento da área).

A sintaxe destas primitivas é :

```
<output> ::= (LINE (<pt>*) <intens> <lintyp> <linwid>) /
             (MARKER (<pt>*) <intens> <mktyp> <mksiz>) /
             (AREA (<pt>*) <styl> <intens>             )
```

onde :

- <pt> ::= POINT
- <intens> ::= 0. / 0.1 / ... / 1.

- <lintyp> ::= SOLID / DOTTED / DASHED
- <linwid> ::= MIN / ... / MAX
- <mktyp> ::= . / + / X / O / *
- <mksiz> ::= MIN / ... /MAX
- <styl> ::= HOLLOW / SOLID / HATCH

- primitivas de entrada : também assentadas sobre primitivas de entrada do núcleo gráfico, permitem ao operador a entrada de dados. Diferentemente das primitivas de saída, cada primitiva de entrada aparece como uma subcélula de diálogo, ao invés de uma primitiva da célula. Assim, são acompanhadas pela palavra reservada SYMBOL que indica sua ativação :

  - SYMBOL LOCATOR;

  - SYMBOL VALUATOR;

  - SYMBOL CHOICE;

  - SYMBOL PICK;

  - SYMBOL STRING.

3.2.2-Pré-Processador e Compilador

O PRE-PROCESSADOR se incumbe de verificar a correção das Células de Diálogo fonte e de realizar o mapeamento das primitivas e das construções lógicas da LED nas respectivas primitivas e construções lógicas do FORTRAN-77. Um compilador FORTRAN-77 é então o responsável pela obtenção das Células de Diálogo objeto.

O PRE-PROCESSADOR é um programa que acessa o arquivo de células de diálogo fonte e lê cada linha de codigo em LED. Atua como um analisador sintático, buscando as palavras reservadas, interpretando-as e identificando então cada construção lógica, cada tipo de dado, enfim, cada primitiva contida na linha. O pre-processador possui uma tabela de correspondências entre as construções lógicas da LED e as do FORTRAN-77 (conforme descrito no item 3.2.1.1) e entre os tipos de dados extendidos e os básicos, podendo então realizar o mapeamento necessário. No caso das construções lógicas este mapeamento é direto, bem definido; já no caso dos tipos de dados, o pré-processador necessita de uma estrutura de dados que sirva para relacionar cada tipo de dado mapeado ao anterior, já que os novos tipos criados em substituição precisam ser referenciados posteriormente cada vez em que se encontrar uma referência aos tipos substituídos.

3.2.3-Supervisor de Execução

O Supervisor de Execução pode agora ser enfocado mais detalhadamente. Para tal, ele é analisado como sendo constituído por três elementos : um Supervisor de "Prototyping", um Selecionador de Sequências de Diálogo e um Gerenciador de Recursos Gráficos.

O Selecionador de Sequências é responsável por, mediante a solicitação da aplicação, determinar quais são (no "prototyping") ou qual é (na execução normal da tarefa aplicação ótima) o(s) subprograma(s) diálogo previsto(s) para aquela interação específica.

Em termos de implementação, o diálogo apresenta-se (conforme pode ser visto na figura 3.2) como uma árvore, onde o Selecionador de Sequências é a raiz. Cada ramo imediatamente abaixo do Selecionador corresponde a uma necessidade de comunicação do programa de aplicação com o operador. Mas, podem ser oferecidas diversas possibilidades de realização de cada uma destas comunicações. Assim, a cada ramo deixando a raiz da árvore pode corresponder uma subárvore, cujos ramos representam as possibilidades de atendimento daquela necessidade de comunicação. A decisão por um ramo ou outro depende de valores de parâmetros passados pelo programa de aplicação ao Selecionador que lhe permitem determinar qual o tipo de necessidade de comunicação e então qual ramo deve ser escolhido.

O Supervisor de "Prototyping" tem sua atividade restrita à fase de "prototyping", já que, em operação normal do diálogo, ele não é necessário. Sua função é a de selecionar ordenadamente, para cada solicitação de interação da aplicação em relação ao operador, cada sequência de diálogo, na forma de subprograma diálogo, provida para atender àquela interação, permitindo a avaliação de todas elas.

Antevendo a implementação, o Supervisor de "Prototyping" agiria como uma estrutura do tipo CASE, segundo a qual, condições seriam manipuladas e testadas implicando no acionamento de uma rotina (subprograma diálogo) para realizar a interação com o operador. Após o retorno do controle de execução ao Supervisor de "Prototyping", as condições são alteradas permitindo o acionamento de outra rotina, e o processo se repete até que todas as rotinas providas para a execução do diálogo correspondente a um dado pedido de interação do programa de aplicação com o operador sejam executadas.

Associando à descrição do Selecionador, o Supervisor de "Prototyping" estaria logo abaixo na árvore e se incumbiria de executar cada ramo da subárvore correspondente ao ramo inicial escolhido pelo Selecionador.

A definição do Gerenciador de Recursos Gráficos, doravante tratado por Gerenciador de Diálogo, surge em

consequência :

- da utilização de recursos de núcleos gráficos padrões pelo diálogo ; tem então o objetivo de adequar os recursos do núcleo gráfico às necessidades do diálogo;

- da necessidade de supervisão dos elementos definidos nas células de diálogo;

- do desejo de proporcionar à aplicação acesso aos recursos do núcleo gráfico.

A primeira consequência vem da necessidade de execução de funções do tipo :

- gerenciamento das transformações de coordenadas das informações gráficas definidas pelo diálogo e mapeadas ao sistema gráfico;

- gerenciamento de segmentos; mapeamento dos elementos de comunicação (ecos e prompt's) aos segmentos gráficos do sistema gráfico, com o necessário provimento da identificação de cada segmento a ser criado;

- gerenciamento do efeito das funções de controle do sistema gráfico para adequá-las às necessidades do diálogo;

- mapeamento dos parâmetros necessários à ativação de primitivas do núcleo gráfico a partir das primitivas gráficas da LED.

Estas funções são comentadas em maiores detalhes no item 3.3.

A segunda consequência atende a funções do tipo :

- atribuição de valores que identificam cada eco, prompt, viewport, picture criados, valores estes que são fornecidos à célula de diálogo nas variáveis utilizadas como nomes destes elementos;

- controle do tempo de vida de cada elemento (eco, prompt). Finda a duração definida, o elemento deve ser explicitamente deletado;

- vinculação das PICTURE's aos elementos eco e prompt correspondentes, conforme definido na célula. Se uma prompt ou eco é alterada, a PICTURE correspondente também deve ser alterada.

- manutenção de informações sobre o desenrolar do diálogo, tais como :

- estado corrente do diálogo : inativo, ativo (existe célula em execução);

- identificação da célula em execução;

- lista das células correntemente ativas (aquela em execução é a mais interna na hierarquia);

- estado de erro (se e onde ocorreu);

- tabela de correlação entre a identificação de um dispositivo gráfico pelo núcleo gráfico e a identificação daquele dispositivo pelo gerenciador de diálogo;

- lista dos elementos VIEWPORT, PROMPT, ECHO e PICTURE definidos, contendo o identificador, o tipo de elemento, os atributos correntes e a células em que cada elemento está definido.

A terceira consequência tem a ver com o oferecimento de facilidades gráficas à aplicação. Pode ser interessante permitir à aplicação acesso às facilidades gráficas do núcleo que dá suporte ao Processador de Diálogo. Ocorre, entretanto, que restrições devem ser feitas a este acesso, para que não haja conflitos de acesso entre a aplicação e a camada de diálogo, o que prejudicaria a representação final. A solução apresenta-se como sendo uma solução uniforme para a utilização do núcleo gráfico por parte tanto do diálogo quanto da aplicação, através da definição de uma primitiva do gerenciador de diálogo para cada recurso do núcleo gráfico, a ser mapeada a estes recursos após tratamento das restrições pelo Gerenciador. No item 3.3 são detalhados alguns dos principais tratamentos a serem efetuados pelo gerenciador às facilidades do núcleo gráfico.

3.2.4-Utilitários de "Prototyping"

Como utilitários de "prototyping" podem ser definidos inúmeros subprogramas para avaliação do desempenho de cada sequência de diálogo. Utilitários de propósito geral, ou seja, que poderiam ser interessantes para avaliação de protótipos para qualquer tipo de aplicação, podem ser exemplificados como sendo de Coleta de Dados, de Monitoramento e de Análise.

3.2.4.1-Coletor de Dados

É responsável pela coleta de dados estatísticos sobre a interação.

São dados de interesse :

- tempo de resposta do operador a um pedido de dados do sistema;

- tempo de resposta do sistema a uma entrada de dados do operador;

- quantidade de vezes em que o operador fornece dados errados ou inconsistentes;

- tipos de erros cometidos pelo operador;

- quantidade de vezes em que o operador se ressente de uma escolha indevida e manifesta desejo de voltar atras;

- tempos e taxas de transferencia de dados entre o operador e o sistema.

Para a geração de dados de interesse sobre a interação, é necessária a definição de variáveis de controle, acessáveis pelas células de diálogo necessárias e pelo subprograma de coleta de dados, que sejam atualizadas periodicamente e consultadas (ou lidas) pelo COLETOR DE DADOS quando necessário.

Para saber quantas vezes uma rotina foi executada, basta colocar nela uma variável sendo incrementada. A qualquer momento, a variável contém a quantidade de vezes em que a rotina foi executada. Para medir tempos de resposta entre operador e sistema, basta que quando um dos parceiros na comunicação solicite dados, seja ativado um relógio que, ao ser desativado quando da obtenção da resposta pelo outro parceiro, represente o tempo de resposta. Medir a quantidade de vezes em que o operador solicita informações ("help") ao sistema e a quantidade de vezes em que ele se ressente de uma escolha pode ser feito de forma similar.

3.2.4.2-Monitor

O MONITOR pode utilizar dados coletados para três tipos de realimentação :

- ao operador para cientificá-lo do decorrer da interação; é possível ao autor de diálogo especificar ao Processador como e onde ele deseja que esta informação seja dada ao operador;

- a um subprograma de análise para fornecer dados para a análise de desempenho;

- criação de um "journal file", documentando o desenrolar da sessão de trabalho, para fins de análise posterior ao seu término.

3.2.4.3-Analisador

Elemento chave na operação de otimização do diálogo (adaptação ao operador), pode analisar o seu desenrolar e decidir por alterações nas descrições de diálogo ou pela substituição de um conjunto de descrições por outro.

Pode-se ter o Analisador operando em dois modos, um "on-line" em relação ao "prototyping" e outro "off-line".

Aplicando estatística aos dados coletados sobre a interação, o analisador pode fazer três tipos de inferências :

- USO ERRADO DO SISTEMA : causado pela utilização de forma indevida de algumas seqüências de diálogo. Sua correção exige o oferecimento de novas alternativas, através da programação de novas Células de Diálogo;

- DIALOGO INCOMPLETO : quando operadores identificam seqüências de diálogo como sendo muito complicadas ou ineficientes. Exige a criação de Células adicionais ou a alteração de Células já existentes;

- MAU PROJETO : exige o reprojeto e a reprogramação do diálogo, através da criação de um novo conjunto de Células de Diálogo.

3.2.5-Limpador

O Limpador é um programa que acessa os arquivos de células de diálogo fonte, processa, ou edita, as células eliminando do código todas as primitivas de ativação de FUNCTION's, não mais necessárias após a obtenção do diálogo ótimo para uma determinada aplicação.

3.3-IMPLICAÇÕES DO USO DE UM NÚCLEO GRÁFICO PADRÃO COMO SUPORTE A SISTEMAS DE TRATAMENTO DE DIÁLOGOS

Como discutido no item 2.3, a utilização de recursos gráficos proporciona um aumento de efetividade do diálogo usuário-computador. Ainda no item 2.3, justificou-se a adoção de um núcleo gráfico padrão para dar suporte à geração e manipulação das figuras a serem utilizadas para esta comunicação. Entre as vantagens enumeradas para esta adoção, destaca-se a independência do núcleo gráfico padrão em relação a linguagem, aplicação e dispositivo específicos.

Por outro lado, foi também apontada no item 3.2.3 a necessidade da existência de um elemento Gerenciador de Diálogo que se responsabilize pela adequação dos recursos do núcleo gráfico às necessidades do diálogo e da aplicação (caso seja permitido à aplicação o uso de facilidades gráficas). Algumas das funções a serem desempenhadas pelo gerenciador de diálogo têm a ver com a supervisão dos elementos definidos na camada de diálogo e dependem então da linguagem e dos mecanismos particulares de representação de estruturas de diálogos do Processador de Diálogo. Outro é o caso das funções do gerenciador dependentes das características específicas dos núcleos gráficos padrões, ou seja, a adoção de núcleos gráficos padrões como suporte a sistemas de tratamento de diálogos traz algumas implicações para o gerenciador de diálogo que independem do método utilizado por estes sistemas para a representação da estrutura do diálogo. Assim, neste item serão analisadas em detalhe implicações do uso de núcleos gráficos padrões (em alguns casos, particularizados para a norma GKS, em virtude de sua aceitação como padrão internacional) por sistemas de diálogo. Algumas destas implicações já foram enfocadas na literatura científica (/BORU 81/, /GIIT 83/), mas não extinguem os tipos possíveis e, além de não terem sua solução apontada, não foram ainda extensivamente discutidos.

O primeiro tipo de implicação ocorre quando do oferecimento de recursos gráficos à aplicação e tem a ver com inconsistências que seriam provocadas pelo acesso aos recursos gráficos por parte da aplicação, se este acesso ocorresse sem o conhecimento do gerenciador de diálogo. Esta consequência pode ser contornada através da definição de primitivas do gerenciador de diálogo, a serem mapeadas às funções gráficas pelo gerenciador, que se incumbe de impor as restrições necessárias. São exemplos de restrições :

- a aplicação determina a criação de um segmento com um nome sendo utilizado pelo diálogo. Para contornar o conflito, o gerenciador de diálogo deve substituir o nome dado pela aplicação por um nome criado pelo próprio gerenciador e armazenar a correspondência entre os nomes para referências futuras;

- a aplicação deseja criar um segmento e existe um, do diálogo, aberto. Quando a aplicação necessita abrir um segmento e ativa a primitiva correspondente, se não há segmento aberto, o gerenciador de diálogo ativa a função OPEN SEGMENT do núcleo gráfico. Se, por outro lado, há segmento aberto, o gerenciador necessita de recursos para armazenar o estado corrente do sistema e restabelecê-lo posteriormente. Para reabrir o segmento então fechado, existem duas possibilidades : criar um segmento lógico, no diálogo, correspondente a diversos segmentos do núcleo gráfico ou implementar uma função REOPEN SEGMENT em extensão ao núcleo gráfico. Uma melhor opção seria a utilização do recurso apresentado pelo próprio núcleo gráfico de inserir segmentos. Assim, após o segmento do diálogo ser fechado e armazenado, é atendida a solicitação da aplicação, findo o que, é aberto um novo segmento do diálogo e inserido nele o segmento anterior, que é então deletado e cede seu nome ao segmento novo;

- a função CLEAR, quando acionada pela aplicação, não deve deletar dados relevantes ao diálogo, necessitando então de um tratamento especial para eliminar somente informação gerada pela aplicação. Para não alterar a função da norma gráfica, pode-se utilizar o recurso de redesenhar as partes referentes ao diálogo, o que pode ser feito sob o controle do gerenciador de diálogo. Deve-se observar que figuras no diálogo sempre correspondem a segmentos do núcleo gráfico, ficando então armazenadas em seu interior;

- a função REDRAW ALL SEGMENTOS ON WORKSTATION; particular da norma GKS, só deve redesenhar figuras geradas pela aplicação e não elementos do diálogo, sob pena de perder informação relevante ao diálogo. O Gerenciador deve verificar quais segmentos não pertencem ao diálogo e somente estes devem ser deletados e redesenhados;

- a função SET DEFERRAL STATE, particular do GKS, controla a visibilidade de modificações de figuras em relação aos processos de entrada. Ela deve ser gerenciada pelo diálogo para que sua ativação não interfira indevidamente nos efeitos das funções de saída e de atributos, que são controladas pelo diálogo;

- as funções de saída só devem ser oferecidas à aplicação via primitivas do gerenciador de diálogo, para evitar interferência com a saída relativa às prompt's e ecos do diálogo;

- com relação às funções de transformação oferecidas pelo GKS, observa-se que as transformações de workstation, se utilizadas pela aplicação, fatalmente comprometem dados relevantes ao diálogo, já que as viewports de workstation (desconhecidas da aplicação) são definidas pelo diálogo em

concordância com o layout de tela pretendido pelo diálogo. A transformação de normalização entretanto, pode ser utilizada pela aplicação, desde que o gerenciador de diálogo reserve a si a responsabilidade de indicar o posicionamento da viewport no espaco NDC, inclusive inviabilizando as dimensões solicitadas caso este posicionamento não possa ser feito de forma a resguardar os elementos de diálogo. Deve-se observar que se a aplicação puder especificar livremente o posicionamento de sua informação gráfica no espaço NDC, pode sobrepor esta informação à informação relevante ao diálogo e, quando da transformação de workstation, mesmo sendo esta transformação realizada sob controle do diálogo, o gerenciamento de informação na tela da workstation está comprometido. Em princípio, tranformações de segmentos para segmentos criados pela aplicação não necessitam de restrições. De qualquer forma, o gerenciador do diálogo pode se utilizar de funções INQUIRE GKS para acessar as matrizes de transformação e analisar as consequências de sua aplicação. Com relação às funções de transformação oferecidas pela norma CORE, raciocínio similar pode ser feito já que, apesar das diferenças entre as duas normas com relação aos espaços de coordenadas, o efeito final é basicamente o mesmo. A prática pode demonstrar, entretanto, que este gerenciamento de window (controle das áreas de exibição dos elementos gráficos) pode se tornar complexo para ser realizado pelo gerenciador do diálogo sem causar impactos ou modificações na estrutura do Núcleo Gráfico. Vale notar também, que mesmo que não sejam colocadas disponíveis à aplicação as funções de transformação, ou parte delas, é necessário ao diálogo um gerenciamento de window para traduzir as requisições do projetista de diálogo em termos de recursos do núcleo gráfico;

- as funções de entrada não podem estar disponíveis à aplicação. Entrada é responsabilidade do diálogo e deve ser executada sempre via camada de diálogo (através de células de diálogo).

Em resumo, este primeiro tipo de implicação exige a definição de primitivas do gerenciador de diálogo a serem mapeadas, após tratamento individualizado pelo gerenciador, às primitivas do núcleo gráfico, evitando então interferências no efeito das funções do núcleo gráfico, e mantendo a portatilidade do gerenciador em relação a implementações de núcleo gráfico. Deve-se notar que estas primitivas serão utilizadas pelo programa de aplicação na forma de chamadas de subrotinas (como se o programa estivesse realmente acionando as funções oferecidas pelo núcleo gráfico), o que leva à necessidade de se implementar rotinas intermediárias que sejam acionadas pelo programa de aplicação e reflitam o tratamento devido ao gerenciador de diálogo antes de acionarem as rotinas correspondentes oferecidas pelo núcleo gráfico de apoio. Vale notar também que estas primitivas devem ser as mesmas utilizadas pela Linguagem de Especificação de Diálogos, para evitar a duplicação de primitivas para cada

facilidade oferecida pelo núcleo gráfico, devendo ser definida uma chave para indicar se a primitiva foi acionada pelo diálogo ou pela aplicação.

Um segundo tipo de implicação surge da relação, existente no núcleo gráfico padrão, entre a entrada e a saída gráficas. Inicialmente, o modelo de tais núcleos gráficos considerava entrada e saída completamente separadas. Assim, não se podia gerar saída gráfica diretamente a partir da ação do operador, que é uma ação de entrada. Isto impedia então a geração de realimentação automática ao comando dado pelo operador. Nos refinamentos mais recentes do GKS, surgiu um recurso para atender a esta demanda : o tipo de prompt/eco (PROMPT/ECHO TYPE) que permite a geração automática de prompt's e ecos sob o controle do núcleo gráfico, mas sem a utilização das primitivas gráficas de saída, ou seja, contando com os próprios recursos de cada dispositivo para a geração de prompt's e ecos. Ficam cadastrados no núcleo os tipos válidos para cada dispositivo e, através de funções oferecidas pelo núcleo, o tipo escolhido pode ser selecionado. Esta solução, entretanto, não atende completamente ao modelo proposto para os gerenciadores de diálogo, segundo o qual quem define e controla a geração de prompt's e ecos é o gerenciador (já que prompt's e ecos estão afetos ao diálogo) e não o núcleo gráfico e estas prompt's e ecos devem ser geradas via primitivas de saída do próprio núcleo gráfico. A figura 3.4 apresenta o modelo de entrada e saída no GKS e o modelo necessário para o gerenciador e o núcleo gráfico operarem em conjunto (/GIIT 83/).

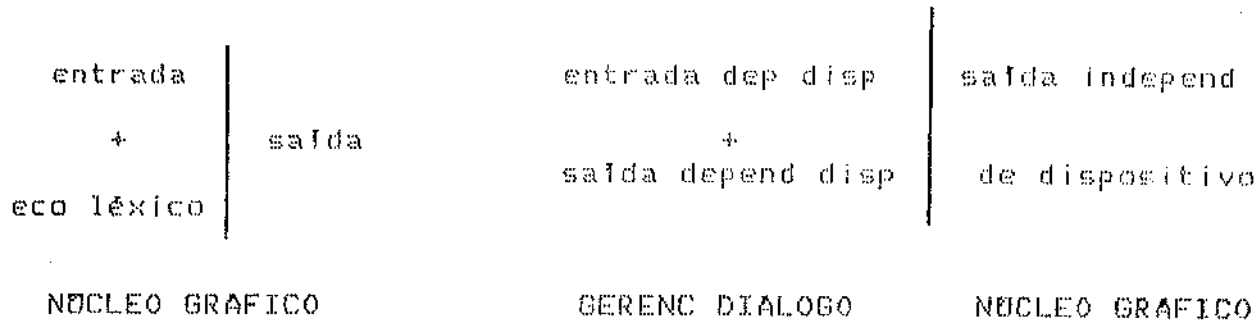


fig. 3.4 : Modelos para o núcleo gráfico e o gerenciador de diálogo.

Este problema não é de fácil solução sem alterações na implementação do núcleo gráfico. Os tipos de prompt/echo controlados pelo núcleo gráfico que realizarem somente realimentação no nível léxico (do tipo "mensagem recebida"), como por exemplo o movimento do cursor na tela acompanhando o

movimento do dispositivo de entrada locator, e que não interfira na realimentação no nível semântico (do tipo "mensagem entendida") controlada pelo gerenciador de diálogo, podem coexistir pacificamente. Estes tipos podem ser selecionados pelo gerenciador através das funções de set de tipo de prompt/eco oferecidas pelo núcleo gráfico. Aqueles tipos de eco próprios do dispositivo que tenderem a prejudicar o efeito do eco realizado pelo gerenciador de diálogo, entretanto, devem ser inibidos por este gerenciador. Tal procedimento não envolve simplesmente alterações no núcleo gráfico, podendo chegar a afetar a implementação do driver entre o núcleo gráfico e o dispositivo.

Em resumo, este segundo tipo de implicação envolve não só a filosofia da implementação do núcleo gráfico como também da implementação dos mecanismos de prompt/eco nos dispositivos. É um problema de difícil solução sem afetar as filosofias de ambos o núcleo gráfico e o gerenciador de diálogo (enquanto definido sobre um núcleo gráfico).

Como terceiro tipo de implicação pode ser citado o conceito de dispositivos virtuais de entrada, premissa básica nos núcleos gráficos padrões. Dois aspectos de importância :

- o comportamento dos dispositivos lógicos é consistente quando visto pelo programador, ou seja, os dispositivos só diferem no tipo de dado retornado e em detalhes da geração de prompt's e ecos. Esta consistência beneficia o programador pois simplifica, ou normaliza, o acesso a uma gama de dispositivos de entrada;

- por outro lado, o programador não distingue entre diferentes dispositivos físicos que retornam o mesmo tipo de dado (por simulação ou não). Ocorre que muitas vezes são as características específicas de cada dispositivo que contribuem para a boa performance da interface com o usuário, mesmo que para efeito do núcleo gráfico tais diferenças não sejam relevantes. Além disso, o fato destas características específicas estarem isoladas do programador impede que dispositivos particularmente excelentes para interações complexas com o operador tenham sua capacidade funcional completamente explorada pelo programador.

Em resumo, o conceito de dispositivos lógicos por um lado beneficia o utilizador do núcleo gráfico, mas, com relação à performance da interface usuário, pode ter um efeito negativo. Tal problema não tem solução, já que este conceito é básico na filosofia do núcleo gráfico e não pode ser alterado sem prejudicar todo o desenvolvimento do padrão.

Um quarto tipo de implicação ocorre devido à desproporcionalidade existente entre a quantidade de recursos

oferecidos pelo núcleo gráfico e a quantidade de recursos a serem utilizados pelo gerenciador de diálogo. Por um lado, o gerenciador por vezes se vê obrigado a executar funções de controle gráfico no topo do núcleo gráfico e por vezes se vê obrigado a desprezar recursos oferecidos pelo núcleo gráfico ou até em utilizar recursos que não aumentam seu potencial de atuação. Como exemplo do primeiro ponto, tem-se a reabertura de segmentos citada acima e como exemplo do último ponto, a obrigatoriedade de ativação de funções irrelevantes ao diálogo mas exigidas pelo núcleo gráfico, como por exemplo, OPEN e CLOSE GKS e OPEN, ACTIVATE, DEACTIVATE e CLOSE WORKSTATION o são na norma GKS. No mapeamento entre recursos oferecidos e necessários, devem ser utilizados somente os recursos estritamente necessários, para não tornar o sistema executável extenso. Cada norma gráfica define níveis válidos de implementação (ver item 2.3). Assim, deve ser tomado aquele nível que mais aproxime em recursos das necessidades do gerenciador. Por outro lado, também pode ser definido um nível intermediário, o que certamente implica em perda de portatilidade mas pode trazer uma adequação melhor do núcleo gráfico ao gerenciador de diálogo.

Em resumo, este quarto tipo de implicação não apresenta solução, já que é uma decorrência direta da opção de utilização do núcleo gráfico e não pode ser contornado de outra forma que não por uma das duas apontadas.

Um quinto tipo de implicação decorre da dificuldade de se ter o gerenciamento de diversos diálogos simultâneos em uma mesma estação de trabalho. Esta necessidade vem, por exemplo, de situações encontradas em linhas de montagem de indústrias, onde sistemas de processamento distribuído executam diversas tarefas simultâneas e um supervisor se utiliza de um terminal para verificar o andamento de cada uma das tarefas, podendo inclusive interferir na sua realização, enviando mensagens aos operadores locais ou acionando procedimentos de emergência. Este operador supervisor necessita de ou definir uma área na tela do terminal para cada diálogo com as diversas tarefas, ou uma tela virtual completa para cada um destes diálogos, representando a tela física do terminal um conjunto de camadas virtuais independentes, acessáveis independentemente e tendo seu controle transferido ao operador. Os núcleos gráficos padrões não oferecem diretamente esta facilidade, devido ao fato de só permitirem o endereçamento de uma única superfície de visualização. Assim, esta tarefa deve ser realizada pelo gerenciador, que pode por exemplo utilizar os recursos de segmentação oferecidos pelos próprios núcleos gráficos padrões.

Em resumo, esta restrição não encontra solução pois vai de encontro a um dos principais pontos de apoio da definição dos padrões gráficos.

Finalizando, pode-se reafirmar que as implicações citadas não extinguem a gama de tipos a serem defrontados com a utilização de núcleos gráficos padrões como suporte aos sistemas de tratamento de diálogos. Assim, a decisão de adotar ou não o suporte do núcleo gráfico é dependente do peso das implicações comentadas e de considerações do tipo :

- abrangência pretendida para o sistema de tratamento de diálogos : que tipos de aplicações, de instalações, etc.;

- oferecimento ou não de facilidades gráficas às aplicações;

- disponibilidade ou não de ferramentas gráficas mais apropriadas ao tratamento exigido pelo diálogo do que aquelas oferecidas pelos núcleos gráficos padrões;

- necessidade ou desejo de se manter portatilidade do gerenciador de diálogo em relação a implementações de núcleo gráfico.

Vale notar que a literatura técnica já apresenta trabalhos onde é discutida a utilização ou definição de padrões gráficos mais apropriados para a realização de diálogos operador-aplicação, principalmente no tocante ao gerenciamento de recursos gráficos e à realização de entrada gráfica.(/ROSE 83/)

4-CONCLUSÕES

A fase inicial do trabalho foi dedicada ao estudo de literatura na área de computação gráfica, especialmente no tocante à interação homem-máquina e aos esforços de padronização de software gráfico (CORE, GKS, IGES, CGI, CGM, UIMS, etc).

Paralelamente, foram realizados diversos trabalhos práticos relacionados aos temas estudados, como por exemplo :

- implantação de uma implementação da versão 6.2 da norma GKS (realizada na Universidade de Darmstadt, na Alemanha Federal) nos três sistemas de computação disponíveis : PDP-11, PDP-10 e VAX-780 (/SILV 85a/, /SILV 85b/);

- implementação de interfaces de entrada/saída (/SILV 85c/) entre o núcleo e dispositivos terminais alfanumérico, semigráfico e gráfico (GT40, terminal gráfico vetorial regenerativo), e de programas de teste, inclusive um monitor para uso interativo das capacidades do sistema gráfico então obtido. Foi necessário o estudo das capacidades de cada dispositivo e das técnicas de interação e de simulação, por software, de classes lógicas de entrada por dispositivos de outras classes, para se encontrar a melhor forma de mapear as primitivas de entrada e de saída do núcleo gráfico às capacidades disponíveis em cada periférico;

- participação na implementação de um núcleo gráfico segundo a versão 7.2 da norma GKS o que trouxe familiaridades com os refinamentos surgidos na norma gráfica.

Uma vez realizado este estudo básico de literatura, apoiado nas diversas experiências práticas já comentadas, o que permitiu a geração do capítulo 2 do trabalho, iniciou-se a tarefa de formalização de uma ferramenta para auxílio à geração de sistemas interativos. Foram pesquisados na literatura diversos sistemas desenvolvidos com propósito semelhante. Foram também estudados os fatores humanos que influem no comportamento de operadores humanos.

Definiu-se então, conforme apresentado no capítulo 3, o sistema Processador de Diálogo como um elemento voltado ao auxílio de projeto de sistemas interativos. Para maior clareza, definiram-se três fases de atuação do Processador de Diálogo : definição, "prototyping" e geração.

Na fase de definição, através de uma Linguagem de Especificação de Diálogos, baseada no método de representação por Células de Diálogo, diálogos podem ser definidos, editados, pré-processados e compilados. É então obtida uma tarefa aplicação envolvendo o programa de aplicação com diversos subprogramas diálogo para cada interação, cada subprograma realizando esta

interação de uma forma diferente.

Na fase de "prototyping" (teste de protótipos de diálogo), a tarefa aplicação pode ser executada, podendo ser coletados e avaliados dados relativos ao desempenho de cada subprograma diálogo e executados subprogramas de apoio à avaliação do diálogo. Como resultado da avaliação, alterações podem ser efetuadas em uma ou algumas das sequências de diálogo, objetivando o seu aprimoramento. Novas execuções e consequentes avaliações podem ser realizadas. Por outro lado, entre as diversas possibilidades de realização de cada interação do operador com o programa de aplicação, pode ser escolhida a de maior efetividade. As funções de apoio ao "prototyping" podem ser providas pelo implementador do Processador de Diálogo ou desenvolvidas pelo projetista de interação, mas são executadas via primitiva oferecida pela LED.

Na fase de geração, uma vez determinada na fase anterior a melhor configuração para a tarefa aplicação, ou seja, para cada interação a se realizar qual a sequência de diálogo apropriada, são subtraídos todos os componentes do Processador de Diálogo não necessários à utilização do diálogo e é gerada uma tarefa aplicação otimizada.

Ao longo do trabalho, diversas vantagens da utilização da ferramenta, então definida, foram apresentadas. Por outro lado, sempre que necessário, os pontos de conflito surgidos entre os elementos envolvidos foram analisados. As vantagens de uso do Processador de Diálogo podem ser resumidas como :

- formalizar, ou padronizar, a arte de projetar diálogos. Um tratamento padronizado possibilita o rápido aprendizado da técnica;

- oferecer uma Linguagem de Especificação de Diálogos clara, legível e objetiva, mapeada sobre uma linguagem padrão, largamente conhecida e empregada, o FORTRAN-77;

- permitir testar diversos diálogos, compará-los e selecionar o de melhor efetividade, adaptando então o diálogo a um operador e a uma aplicação específicos;

- permitir projeto modular, com a utilização de um ação interdisciplinar, onde cada camada do sistema interativo fica a cargo do especialista correspondente;

- possibilitar a utilização de informações gráficas no diálogo entre operador e aplicação, o que traz aumento de efetividade;

- permitir economia de custos, pois o Processador pode ser utilizado por diversas aplicações. A mesma interface usuário pode ser utilizada por mais de uma aplicação, o que justifica um esforço na depuração da interface que aumente sua confiabilidade;

- permitir explorar características de dispositivos e técnicas de interação com o objetivo de tornar a interface "amigável" ao usuário.

e quanto aos pontos de conflito detectados limitaram-se à relação entre a camada de diálogo e o núcleo gráfico padrão, o que implicou na definição de um Gerenciador de Diálogo.

A decisão de utilização do núcleo gráfico padrão como apoio ao Processador de Diálogo veio do desejo de se ter uma interface com o mundo exterior portátil e consistente. Desta decisão puderam ser detectadas, durante o trabalho, diversas vantagens resumidas como :

- possibilitar que o Processador seja definido independente (portátil) em relação à linguagem de aplicação, ao computador hospedeiro, aos periféricos e ao tipo de aplicação. Esta independência possibilita a substituição do hospedeiro ou de periféricos sem impactos no sistema interativo então construído. Possibilita também a execução de alterações na camada de aplicação sem impacto na camada de diálogo;

- facilitar a implementação do Processador Diálogo, tanto pelo fato de se poder adotar uma implementação de núcleo gráfico já existente, como pelo fato de se ter uma norma que define completamente a filosofia do núcleo gráfico, dispensando esforço na confecção de um pré-projeto do núcleo gráfico;

- dispensar a definição de recursos para geração e manipulação de figuras dentro do Processador de Diálogo.

Por outro lado, foi detectada a necessidade de se definir um elemento Gerenciador de Diálogo (ou Gerenciador de Recursos Gráficos), que se responsabilize por supervisionar a utilização dos recursos do núcleo gráfico pela camada de diálogo, adequando os recursos do núcleo gráfico às necessidades da camada de diálogo, e para permitir às aplicações que se utilizam do Processador de Diálogo, o acesso às mesmas facilidades gráficas oferecidas ao diálogo. Assim, o núcleo gráfico foi analisado sob o ponto de vista de atender ao diálogo e à aplicação e foram identificadas as seguintes implicações quando de seu uso :

- o oferecimento de facilidades gráficas à aplicação, tendo a vantagem de aumentar o espectro de atuação do Processador, provoca a definição de primitivas do gerenciador de diálogo a serem mapeadas, após tratamento, às primitivas do núcleo gráfico ;

- o modelo entrada-saída existente no núcleo gráfico vai contra o modelo necessário ao gerenciador de diálogo, no qual prompt's/echo's são controlados pelo gerenciador. Este

problema exige a avaliação dos tipos de prompt/echo intrínsecos aos dispositivos com relação ao seu efeito positivo ou negativo na geração controlada pelo gerenciador. Os tipos que forem prejudiciais devem ser inibidos;

- o conceito de dispositivos lógicos de entrada, que beneficia o projetista de interação no tocante à consistência de comportamento, mas que pode prejudicar a efetividade de interação ao impossibilitar a exploração de capacidades específicas de dispositivos mais sofisticados. É um problema sem solução, por envolver um conceito intrínseco ao núcleo gráfico;

- a desproporcionalidade entre os recursos oferecidos pelo núcleo gráfico e os necessários ao Processador de Diálogo. Além das possibilidades de adotar um nível de implementação do núcleo gráfico mais adequado ou de definir um novo nível, não há possibilidade de resolução por ser decorrência direta da utilização do núcleo gráfico;

- a impossibilidade de gerenciamento de diversos diálogos simultâneos, principalmente pela incapacidade do núcleo gráfico de permitir endereçamento simultâneo de diversas superfícies de visualização num mesmo dispositivo.

Em linhas gerais, o Processador de Diálogo está constituído pelos seguintes elementos :

- PRÉ-PROCESSADOR : é o elemento que se incumbe do tratamento das células de diálogo fonte, mapeando as características extendidas da LED às características básicas da linguagem FORTRAN-77. Sua função é de produzir rotinas em FORTRAN-77 que, ao serem submetidas ao compilador desta linguagem, produzem células de diálogo objeto;

- SUPERVISOR DE EXECUÇÃO : é responsável pelo interfaciamento da camada de diálogo com a camada de núcleo gráfico. Compõe-se de três partes :

  - SUPERVISOR DE "PROTOTYPING" : tem sua função restrita ao "prototyping", sendo responsável por fazer com que todas as sequências de diálogo (subprogramas diálogo) providas para atender a uma dada interação, sejam executadas uma por vez, o que permitirá a avaliação de cada uma delas;

  - SELECIONADOR DE SEQUENCIAS : sua função é de determinar para cada solicitação de interação do programa de aplicação com o operador o conjunto de subprogramas diálogo provido para atender àquela interação;

  - GERENCIADOR DE DIALOGO : também tratado por

Gerenciador de Recursos Gráficos, é responsável por adequar os recursos oferecidos pelo núcleo gráfico às necessidades da camada de diálogo e por supervisionar a utilização de facilidades gráficas pela aplicação;

- UTILITARIOS DE "PROTOTYPING" : são subprogramas que dão apoio à realização do "prototyping", auxiliando na avaliação das diversas estruturas de diálogo. Estes utilitários podem ser providos pelo implementador do Processador de Diálogo ou pelo próprio projetista de interação, e devem ser capazes de executar funções do tipo de coleta de dados, monitoramento, análise de desempenho, etc., e sua execução é comandada através de uma primitiva oferecida pela LED;

- LIMPADOR : é o elemento que acessa as células de diálogo escolhidas como sendo as de maior efetividade e elimina delas todas as primitivas relacionadas ao "prototyping", permitindo a geração da tarefa aplicação ótimizada.

Não foram tratados diversos problemas, como por exemplo, o gerenciamento dos dados envolvidos com o Processador de Diálogo e o gerenciamento dos diversos métodos de aplicação e de diálogos. Entretanto, o objetivo maior deste trabalho foi alcançado, ao definir funcionalmente o elemento Processador de Diálogo, o que permitirá a obtenção deste elemento através de trabalhos futuros, responsáveis pela implementação de cada um dos componentes definidos.

Podem ser citadas como propostas para a continuação deste trabalho :

- detalhamento do Supervisor de Execução, englobando os elementos Gerenciador de Diálogo, Supervisor de "Prototyping" e Seletor de Sequências de Diálogo;

- implementação dos elementos detalhados;

- implementação da Linguagem de Especificação de Diálogos, através do desenvolvimento do Pré-processador;

/ARNO 84/ - Arnold, D.B.; "Report on the ANSC X3H3 Meeting at Carmel - OCT 1st-5th'84 - Computer Graphics Forum - 3 (1984) - pp 321/322

/BERG 78/ - Bergeron, R.D.; Bono, P.R.; Foley, J.D.; "Graphics Programming Using the CORE SYSTEM" - Computing Surveys - 10 - 4 - Dec 78 - pp 389/443

/BLAC 77/ - Black, J.L.; "A General Purpose Dialogue Processor" National Computer Conference - 1977 - pp 397/408

/BONO 82/ - Bono, P.R.;Encarnacao,J.L.;Hopgood,F.R.A.;ten Hagen, P.J.W.; "GKS - The First Graphics Standard" - IEEE Computer Graphics and Applications - vol 2 - n. 5 JULY 1982 - pp 9/23

/BORU 80/ - Borufka, H.G.; Pfaff, G.; "The Design of a General Purpose Command Interpreter for Man-Machine Communication" - Proceedings IFIP WG 5.2/5.3 Conference in Tokyo - 1980

/BORU 81/ - Borufka, H.G.;ten Hagen,P.J.W.;Kuhlmann, H.W.;Weber, H.R.;"On Defining Interactions by Dialogue Cells" Fachgebiet Graphisch Interaktive Systeme -Technische Hochschule Darmstadt - GRIS 81-7 - Germany

/BORU 82/ - Borufka, H.G.;ten Hagen, P.J.W.;Kuhlmann, H.W.; "Dialogue Cells :A Method for Defining Interactions" - IEEE Computer Graphics and Applications - vol 2 - n. 5 - JULY 1982 - pp 25/33

/BOS 77/ - Van Den Bos,J. et al; "GPGS : a Device Independent General Purpose Graphics System for Stand-Alone and Satellite Graphics" - Proceedings 1977 SIGGRAPH Conf - 11 - 2 - SUMMER 1977 - pp 112/119

/CUNH 81/ - Cunha, J.D.; "Sistemas Gráficos Interactivos Configuráveis para Diferentes Aplicações" - Tese apresentada à concurso para especialista do Laboratório Nacional de Eng. Civil - Lisboa - 1981

/DEHN 81/ - Dehning, W.; Essig, H.; Maass, S.; "The Adaptation of Virtual Man-Computer Interfaces to User Requirements in Dialogs" - Lecture Notes in Computer Science n. 110 - Edited by G. Goos and J. Hartmanis - Springer Verlag - 1981

/DENE 77/ - Denert, E.; "Specification and Design of Dialogue Systems with State Diagrams"-International Computing Symposium - North Holland Publishing Company - 1977 - pp 417/424

/EASO 80/ - Eason, K.D.; "Dialogue Design Implications of Task Allocation Between Man and Computer" - Ergonomics - 1980 - vol 23 - n 9 - pp 881/891

/EDMO 82/ - Edmonds, E.; "The Man-Computer Interface : a note on Concepts and Design" - International Journal of Man-Machine Studies - 1982 - 16 - pp 231/236

/ENCA 80/ - Encarnacao, J.L. et al;"The Workstation Concept of GKS and the Resulting Conceptual Differences to the GSPC CORE SYSTEM" - Computer Graphics - 14 (3) - 1980 - pp 226/229

/ENCA 82/ - Encarnacao, J.;Hanusa, H;Strasser, W.; "Tools and Techniques for the Description, Implementation and Monitoring of Interactive Man-Machine Dialogues" - 1982 International Zurich Seminar on Digital Communication - Man-Machine Interaction- Proceedings IEEE Cat. n. 82ch1735-0

/ENCA 83/ - Encarnacao, J.;Borufka,H.G.;Hanusa,H.;Kuhlmann, H.W; Pfaff, G.E.;Weber, H.R.;"An Integrated Concept for Generating Graphics Dialogue Processors" - Computer Applications in Production and Engineering - North Holland Publishing Company-IFIP - 1983 - pp 643/658

/ENDE 83/ - Enderle,G.;"GKS-Implementations Overview and Inquiry - Computer Graphics Forum - vol 2 - n 1 - MAR 1983

/ENDE 84a/ - Enderle, G.; GKS-Implementations Overview Second Edition - Computer Graphics Forum - vol 3 - n 2 - JUN 1984

/ENDE 84b/ - Enderle, G.; Kansy, K.; Pfaff, G.;"Comp . Graphics Programming - GKS : The Graphics Standard" - Springer Verlag - 1984

/ENDE 84c/ - Enderle, G.; "The Interface of the UIMS to the Application" - Working Group Report - Computer Graphics Forum - 3 (1984) - pp 175/177

/FOLE 74/ - Foley, J.D.; Wallace, V.L.; "The Art of Natural Graphic Man-Machine Conversation"- Proceedings IEEE - vol 62 - n. 4 - APRIL 1974 - pp 462/471

/FOLE 81/ - Foley, J.D.; Wenner, P.A.; "The George Washington University CORE SYSTEM Implementation" - Computer Graphics - 15 (3) - Aug 1981 - pp 123/131

/FOLE 82/ - Foley, J.D.; Van Dam,A.;"Fundamentals of Interactive Computer Graphics" - Addison Wesley - Reading - Mass - 1982

/FREI 81/ - Freiden, A.; "A Two-dimensional Level 2 CORE SYSTEM for the APPLE II" - Computer Graphics - 14 (4) - MAR

1981 - pp 152/152

/FREI 84a/ - Freitas, C.M.Dal Sasso; "Programação Gráfica Interativa com o PGE/UFRGS" - Relatório Técnico RT 008 - CPGCC- UFRGS - Porto Alegre - RS - jun/1984

/FREI 84b/ - Freitas, C.M.Dal Sasso; "Descrição do Pacote Gráfico PGE/UFRGS" - Relatório Técnico RT 009 - CPGCC- UFRGS Porto Alegre - RS - jul/1984

/GILO 78/ - Giloi, W.K.; "Interactive Computer Graphics - Data Structures, Algorithms,Languages"-Prentice Hall-1978

/GIIT 81/ - "SEATTLE WORKSHOP ON GRAPHICAL INPUT AND INTERACTION TECHNIQUES" - Computer Graphics - 15 (4) - 1981

/GIIT 83/ - GRAPHICAL INPUT INTERACTION TECHNIQUE WORKSHOP SUMMARY - Computer Graphics - 17 (1) - JAN 1983

/GINO 75/ - "GINO-F, the General Purpose Graphics Package Reference Manual" - CAD Centre.- Cambridge - UK - JULY 1975

/GKS 82/ - Information Processing-GRAPHICAL KERNEL SYSTEM(GKS) Functional Description-Draft International Standard ISO/DIS 7942 - ISO TC97/SC5/WG2 N163 - Nov/82

/GKS3 84/ - ISO TC97/SC5/WG2; "Extensions of GKS to 3D - WORKING DRAFT" - N277 - 1984

/GSPC 77/ - GRAPHIC STANDARDS PLANNING COMMITTEE STATUS REPORT ACM/SIGGRAPH - Computer Graphics- 11 (3) - Fall 1977

/GSPC 79/ - GRAPHIC STANDARDS PLANNING COMMITTEE STATUS REPORT ACM/SIGGRAPH - Computer Graphics- 13 (3) - Fall 1979

/GREE 84/ - Green, M.; "Report on Dialogue Specification Tools" - Computer Graphics Forum - 3 (1984) - pp 305/313

/GUED 79/ - Guedj, R.A.; Tucker, H.A.; "Methodology in Computer Graphics" - North Holland - Amsterdam - 1979 - Proceedings of Seillac I Workshop in 1976

/GUED 80/ - Guedj, R.A.;ten Hagen,P.J.W.;Hopgood, F.R.A.;Tucker, H.A.;Duce, D.A.;"Methodology of Interaction" - North Holland - Amsterdam - 1980 - Proceedings of Seillac II Workshop

/GUES 82/ - Guest, S.F.; "The Use of Software Tools for Dialogue Design" - International Journal of Man-Machine Studies - 1982 - 16 - pp 263/285

/HANU / - Hanusa, H.;"Tools and Techniques for the Monitoring of Interactive Graphics Dialogues" - Fachgebiet Graphisch Interaktive Systeme -Technische Hochschule

Darmstadt - Germany

/HANU 82/ - Hanusa, H.; Kuhlmann, H.W.; Pfaff, G.E.; "On Constructing Interactive Graphics Systems" - Proceedings Eurographics 1982 - North Holland - pp 237/248 - 1982

/HEWI 84/ - Hewitt, W.T.; "PHIGS-Programmer's Hierarchical Interactive Graphics System"-Computer Graphics Forum - 3 (1984) - pp 299/300

/HOLL 83/ - Hollnagel, E.; "What we do not Know About Man-Machine Systems" - Internacional Journal of Man-Machine Studies - 1983 - 18 - pp 135/143

/IGES 81/ - "DIGITAL REPRESENTATION FOR COMMUNICATION OF PRODUCT DEFINITION DATA" - AMERICAN NATIONAL STANDARD - ANSI Y14.26M - 1981

/KAIS 82/ - Kaiser,P.;Stetina,I.;"A Dialogue Generator"-Software Practice and Experience - vol 12 - 1982 - pp 693/707

/KASI 82/ - Kasik, D.J.; "A User Interface Management System" - Computer Graphics - 16 (3) - JULY 1982

/MART 73/ - Martin, J.; "Design of Man-Computer Dialogues" - Prentice Hall - 1973

/MART 81/ - Martins, R.P.; "Interface de Entrada para um Núcleo Gráfico"- Tese de Mestrado - UNICAMP - CAMPINAS - SP - 1981

/MICH 78a/ - Michener, J.C.; Van Dan,A.;"A Functional Overview of the CORE SYSTEM with GLOSSARY" - Computing Surveys - 10 - 4 - Dec 1978 - pp 381/387

/MICH 78b/ - Michener, J.; Foley, J.D.; "Some Major Issues in the Design of the CORE GRAPHICS SYSTEM" - Computing Surveys - 10 - 4 - Dec 1978 - pp 445/463

/MOLI 81/ - Molinaro, L.F.R.;"Interface de Saída para um Sistema Gráfico de Base"-Tese de Mestrado-UNICAMP - CAMPINAS - SP - 1981

/MORE 82/ - Moret, B.M.E.; "Decision Trees and Diagrams" - ACM Computing Surveys - vol 14 - n. 4 - DECEMBER 1982 - pp 593/623

/NEWM 78/ - Newman, W.M., Van Dan, A.; "Recent Efforts Towards Graphics Standardization" - Computing Surveys - 10 - 4 - Dec 1978 - pp 365/380

/NEWM 81/ - Newman, W.M. & Sproull, R.F.; "Principles of Interactive Computer Graphics" - 2a. Ed. - McGraw - Hill - 1981

/NICO 81/ - Nicol, C.J.; Kilgour, A.C.; "A Pascal Implementation of GSPC CORE Graphics Package" - Computer Graphics - 15 (4) - Dec 1981 - pp 327/335

/OHLS 78/ - Ohlson, M.; "System Design Considerations for Graphics Input Devices" - Computer - Nov 1978 - pp 9/18

/PARN 69/ - Parnas, D.L.; "On the Use of Transition Diagrams in the Design of a User Interface for an Interactive Computer System" - ACM - Proceedings 24th. National Conference - 1969

/PATA 84/ - Pataca, M.L.M.; "Distribuição de Tarefas entre o Homem e a Máquina em Supervisão de Processos" - Tese de Mestrado - UNICAMP - CAMPINAS - SP - 1984

/PETE 77/ - Peterson, J.L.;"Petri-Nets" - ACM Computing Surveys - vol 9 - n. 3 - SEPT 1977 - pp 223/252

/PHIG 84/ - ANSC X3H3, "PHIGS Functional Description", 84-40 , 1984

/PHIL 75/ - "PHILDIG : General Description" - Phillips public n. UDP-DSA-SCA/75/001/DE/CN - 1975

/REIS 79/ - Reisner, P.;"Using a Formal Grammar In Human Factors Design of an Interactive Graphics System - IBM Research Report RJ 2505-(32755)-San Jose- California

/ROSE 83/ - Rosenthal, D.S.H.; "Managing Graphical Resources" - Computer Graphics - 17 (1) - JAN 1983 - pp 38/45

/ROSS 63/ - Ross, D.T.; Rodriguez, J.E.;"Theoretical Foundations for the Computer-Aided Design System" - Proceedings 1963 AFIPS Spring Jt. Computer Conf - Vol 23 - Spartan Books - Washington - pp 305/322

/SEEH 84/ - Pfaff, G.; Ten Hagen, P. , ed. ; "Seeheim Workshop on User Interface Management Systems" - Springer Verlag - 1984

/SHAC 80/ - Shackel, B.; "Dialogues and Language - Can Computer Ergonomics Help?" - Ergonomics - 1980 - vol 23 - n 9 - pp 857/880

/SILV 85a/ - Silva, M.V.; "Norma GKS (GRAPHICAL KERNEL SYSTEM): Versão 6.2 e Evolução para Versão 7.2" - Relatório Técnico FEC-UNICAMP 24/85 - ABRIL 1985

/SILV 85b/ - Silva, M.V.; "O Sistema KL : Implementação GKS 6.2 - Implantação na UNICAMP" - Relatório Técnico FEC-UNICAMP 25/85 - ABRIL 1985

/SILV 85c/ - Silva, M.V.; "Interfaces de Entrada/Saída para o Sistema KL" - Relatório Técnico FEC-UNICAMP 26/85 - ABRIL 1985

/STEI 84/ - Steinhart, J.E.; "Proposal for GKS Output Level 3 Segment Hierarchy and Editing" - Computer Graphics Forum - 3 (3) - 1984 - pp 289/297

/STEW 80/ - Stewart, T.; "Communicating with Dialogues" - Ergonomics - 1980 - vol 23 - n 9 - pp 909/919

/STLU 82/ - Stluka, F.P.; Saunders, B.F.; Slayton, P.M.; Badler, N.I.; "Overview of the University of Pensylvania CORE SYSTEM Standard Graphics Package Implementation" - Computer Graphics - 16 (2) - Jun 1982 - pp 177/186

/STRA 82/ - Straayer, D.; "Graphics Standards Evolve to Serve Users and Vendors" - Computer Technology Review - Winter 1982 - pp 93/101

/STRU 84/ - Strubbe, H.J.; "Role, Model, Structure and Construction of a UIMS" - Working Group Report - Computer Graphics Forum - 3 (1984) - pp 171/174

/SUTH 63/ - Sutherland, I.E.; "SKETCHPAD : a Man-Machine Graphic Communication System" - Proceedings 1963 AFIPS Spring Jt. Computer Conf - Vol 23 - Spartan Books - pp 329/346

/VDI 84/ - "INFORMATION PROCESSING COMPUTER GRAPHICS VIRTUAL DEVICE INTERFACE" - Functional Description - ANSI X3H3 84/45 - MAR 1984

/VDM 83/ - "DRAFT PROPOSED AMERICAN NATIONAL STANDARD for the VIRTUAL DEVICE METAFILE" - ANSI X3H33 83/16 - ANSI X3H3 83/15 - MAR 1983

/WOOD 71/ - Woodsford, P.A.; "The Design and Implementation of the GINO 3D Graphics Software Package" - Software Pract. Exper. - 1 - OCT 1971 - pp 335

APENDICE : Exemplo de utilização da Linguagem de Especificação de Diálogos.

Com o objetivo de esclarecer os elementos apresentados nos itens 2.2.3 e 3.2.1, inclui-se a seguir um exemplo no qual a Linguagem de Especificação de Diálogos é utilizada na formalização de um projeto de diálogo.

O exemplo descreve uma solução para o diálogo entre uma aplicação e um operador, no qual a aplicação necessita que sejam definidas pelo operador diversas figuras equiláteras. A quantidade de figuras não deve ultrapassar um valor máximo, definido pela aplicação. As figuras podem ser de três tipos : quadrado, triângulo ou losango, possuir dimensões diversas e ser desenhadas com qualquer de três tipos de linha, sólida, tracejada ou ponto-traço.

A aplicação não interessa o armazenamento das figuras e sim a quantidade de figuras criadas, a quantidade de pontos de cada figura, as coordenadas dos pontos e o tipo de linha utilizado pelo operador. A representação visual das figuras serve unicamente de realimentação ao operador, que pode inclusive deletar as figuras cuja representação não tenha correspondido às suas expectativas (a quantidade de figuras deletadas tambem é informada a aplicação).

As opções são apresentadas em menus e a escolha é feita através de uma "light-pen" apontando a tela (simulação de CHOICE).

Para indicar posições para os desenhos, o usuário se utiliza de um "tracking cross", que movimenta um cursor na tela (LOCATOR).

A dimensão das figuras é fornecida pelo teclado (simulando VALUATOR).

Para a identificação de figuras a serem deletadas, é utilizada a light-pen (PICK).

A tela é dividida em quatro áreas dispostas como na figura A.1.

Cada uma das áreas é utilizada conforme abaixo :

- área de trabalho, para geração dos desenhos indicados pelo usuário;

- área para apresentação dos menus de opções permitidas ao usuário;

- área para prompt (pedido de dados ao usuário);

- área para eco (realimentação do sistema, confirmando a

opção feita pelo usuário).

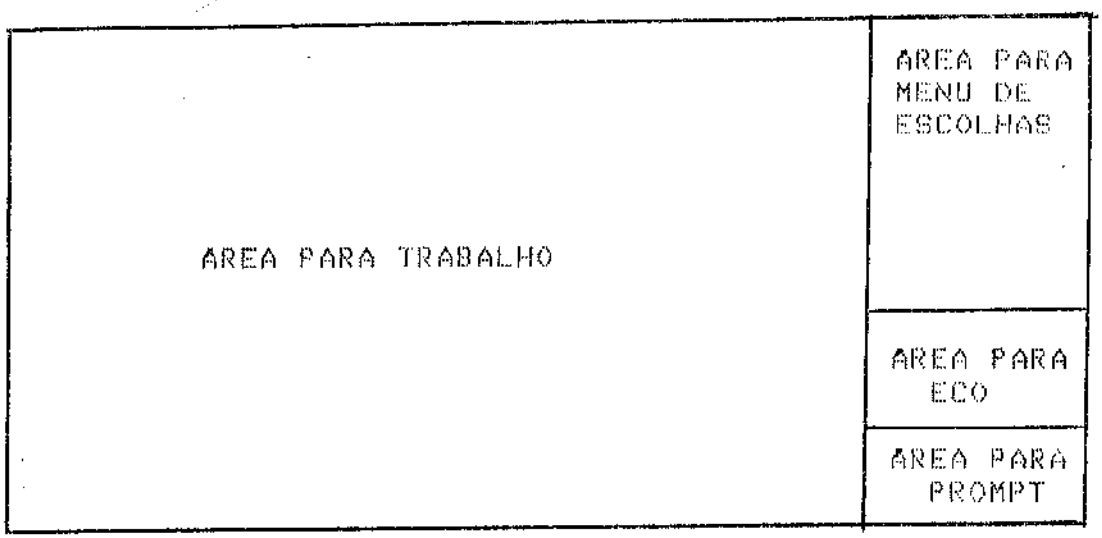


fig. A.1 : Disposição da tela do terminal utilizado para desenho das figuras.

São definidas quatro Células de Diálogo principais e, subcélulas especiais são utilizadas para obter dados do usuário, uma para cada classe lógica de entrada, as quais não são apresentadas por questões de simplicidade. As quatro Células realizam a comunicação com o operador e criam ecos de figuras como forma de apresentar ao operador uma representação visual da figura por ele definida através de um tipo de figura, um ponto inicial, uma dimensão e um tipo de linha. Uma das quatro células, DESENHO, não realiza entrada ou saída mas simplesmente gera as coordenadas adicionais das figuras, a partir do tipo de figura, ponto inicial e dimensão.

A ligação com a aplicação é feita através da célula SISTEMA que recebe da aplicação a quantidade máxima de figuras e devolve à aplicação os dados obtidos.

A célula SISTEMA define os três menus de opções que serão utilizados, o primeiro pela própria célula e os outros dois pela célula CRIAÇÃO e define um ECHO e uma PICTURE, associados, para cada figura a ser desenhada. SISTEMA contem um loop onde o operador é convidado repetidas vezes a fornecer comandos de criação e deleção ou de encerramento (quando o loop é encerrado). Para cada opção um eco é gerado na area de eco e para as duas primeiras opções outra célula é ativada (CRIAÇÃO ou DELEÇÃO), sendo os parâmetros necessários transmitidos. Ao retorno à célula

SISTEMA, um eco de término de operação é gerado.

A célula CRIAÇÃO define prompt's, para solicitar do operador os dados necessários à criação das figuras, e ecos para realimentar os dados ao operador. Menus automaticamente tornados invisíveis (ou seja, cuja duração foi definida como sendo até a proxima prompt na mesma viewport) são modificados para visíveis quando necessário. A célula DESENHO é ativada para fornecer o vetor de coordenadas da figura. Ao final, é criada uma PICTURE correspondendo ao ECHO que representará visualmente a figura criada. A quantidade de figuras é então incrementada de 1.

A célula DELEÇÃO define uma prompt para solicitar a identificação da figura a ser deletada, ativa a célula de entrada de dados, utiliza o identificador da figura para comandar sua deleção, identifica a posição da figura dentro do vetor de identificadores e cancela a quantidade de pontos correspondentes no vetor a ser devolvido a aplicação, incrementando de 1 a quantidade de figuras deletadas.

A célula DESENHO identifica o tipo de figura sendo criada e calcula o vetor de pontos necessário a partir da posição inicial e da dimensão. Para cada figura é acrescentado um ponto igualado ao inicial para gerar a figura fechada, já que é utilizada a primitiva LINE, que é originalmente aberta.

A seguir, são apresentadas as quatro células principais, onde as palavras em letras minúsculas representam nomes de variáveis enquanto as em letras maiúsculas ou são palavras reservadas ou nome de subcélulas. Os comentários são colocados após barra e asterisco (/*). A sintaxe e as características das primitivas utilizadas estão descritas nos itens 2.2.3 e 3.2.1.

```
DIACELL SISTEMA (max,nfig,qtedel,pto,npont,linha)

     IN     INTEGER   max              /* qte máxima de figuras
     TRANS  INTEGER   nfig             /* qte figuras criadas
                      qtedel           /* qte figuras deletadas
                      npont(max)       /* qte pontos por figura
                      linha(max)       /* tipo linha de cada figura
            POINT     pto(max,5)       /* coordenadas dos pontos
     PDECL  END

     LOC    VIEWPORT  janela(4)    /* conjunto de viewports
            PROMPT    menucom      /* menu de comandos
                      menufig      /* menu de tipos figuras
                      menulin      /* menu de tipos linha
            INTEGER   cop          /* operação principal selecionada
            ECHO      ecop         /* eco de oper principal
                      opfim        /* eco de operação concluída
                      echo(max)    /* eco de figura
            PICTURE   quad         /* representação do quadrado
                      triang       /* representação do triângulo
                      losang       /* representação do losango
                      picture(max)/* representação da figura de eco
     ODECL  END

     INIT
          VIEWPORT DEF janela(1) 0.0,0.0,0.8,1.0 DEF END /*trabalho
          VIEWPORT DEF janela(2) 0.8,0.2,1.0,1.0 DEF END /* comando
          VIEWPORT DEF janela(3) 0.8,0.1,1.0,0.2 DEF END /* eco
          VIEWPORT DEF janela(4) 0.8,0.0,1.0,0.1 DEF END /* prompt

          PROMPT DEF menucom VIEW janela(2) TILL NEXT PROMPT SVPT
                       INVISIBLE
                       MENU FOR CHOICE 1
                            1 : 3
                       TEXT (.1,.8)'criar', (.1,.6)'deletar',
                            (.1,.4)'encerrar'
          DEF END

          PICTURE DEF triang INVISIBLE
                  LINE (.1,.7),(.5,.7),(.3,.8),(.1,.7)
                       1 SOLID MAX
          DEF END

          PICTURE DEF quad INVISIBLE
                  LINE (.1,.5),(.5,.5),(.5,.6),(.1,.6),(.1,.5)
                       1 SOLID MAX
          DEF END

          PICTURE DEF losang INVISIBLE
                  LINE (.1,.3),(.3,.2),(.5,.3),(.3,.4),(.1,.3)
                       1 SOLID MAX
          DEF END
```

```
        PROMPT DEF menufig VIEW janela(2) TILL NEXT PROMPT SUPT
                   INVISIBLE
                   MENU FOR CHOICE 2
                        1 : 3
                   PICTURE quad triang lnsang
        DEF END

        PROMPT DEF menulin VIEW janela(2) TILL NEXT PROMPT SUPT
                   INVISIBLE
                   MENU FOR CHOICE 3
                        1 : 3
                   TEXT (.1,.8)'solida', (.1,.6)'tracej',
                        (.1,.4)'pto-trac'
        DEF END

        DO (i=1,max)
             ECHO DEF echo(i) VIEW janela(1) TILL CELLEND
                  PICTURE picture(i)
             DEF END
        FIN

        qtedel=0
        nfig=0
        cop=0
   INIT END

   BODY
        WHILE (cop.NE.3.AND.nfig.NE.max) /*loop de comando

             PROMPT MENU MOD menucom VISIBLE  MOD END
             SYMBOL CHOICE (1,cop)      /*requisita comando

             ECHO DEF ecop VIEW janela(3) TILL NEXT ECHO SUPT
                  SELECT (cop)
                       (1) TEXT (.1,.4)'criar'
                       (2) TEXT (.1,.4)'deletar
                       (3) TEXT (.1,.4)'encerrar'
                  FIN
             DEF END

             SELECT (cop)
                  (1) SYMBOL CRIACAO(max,janela,menufig,menulin,
                                     nfig,npont,linha,pto,picture)
                  (2) SYMBOL DELECAO  (max,janela,qtedel,npont,
                                       echo)
             FIN

             ECHO DEF opfim VIEW janela(3) TILL NEXT PROMPT
                  SELECT (cop)
                       (1)  TEXT (.1,.8)'criacao concluida'
                       (2)  TEXT (.1,.8)'delecao concluida'
                  FIN
             DEF END
        FIN
END DIACELL
```

```
DIACELL CRIACAO (max,janela,menufig,menulin,nfig,npont,linha,pto,
                 picture)

  IN    VIEWPORT   janela(4)     /* conjunto de viewports
        PROMPT     menufig       /* menu tipos figuras
                   menulin       /* menu tipos linha
        PICTURE    picture(max)/* figura de eco
        INTEGER    max           /* qte máxima figuras

  TRANS INTEGER    nfig              /* qtc figuras criadas
                   npont(max)        /* qte pontos por figura
                   linha(max)        /* tipo linha de cada figura
        POINT      pto(max,5)        /* coordenadas dos pontos
  PDECL END

  LOC   PROMPT     tipfig      /* prompt para tipo figura
                   lado        /* prompt para dimensão
                   tiplin      /* prompt para tipo linha
        INTEGER    fig         /* identifi ador tipo figura
        REAL       delta       /* dimensão da figura
        ECHO       ecofig      /* eco de tipo figura
                   ecolin      /* eco de tipo linha
  DDECL END

  INIT
       PROMPT DEF tipfig VIEW janela(0) TILL NEXT ECHO
                  TEXT (,1,,9)' tipo figura/pos inicial'
       DEF END
  INIT END

  BODY

       PROMPT MENU MOD menufig VISIBLE MOD END

       SYMBOL CHOICE (2,fig)        /* seleciona tipo figura

       ECHO DEF ecofig VIEW janela(3) TILL NEXT ECHO SUPT
            SELECT (fig)
                 (1) TEXT (,1,,9)'criacao quadrado'
                 (2) TEXT (,1,,9)'criacao triangulo'
                 (3) TEXT (,1,,9)'criacao losango'
            FIN
       DEF END

       SYMBOL LOCATOR ('tracking cross',pi,janela(1))

       PROMPT DEF lado VIEW janela(4) TILL NEXT ECHO
            TEXT (,1,,9)'entre dimensao'
       DEF END

       SYMBOL VALUATOR ('keyboard',delta)

       PROMPT MENU MOD menulin VISIBLE MOD END
```

```
        PROMPT DEF tiplin VIEW janela(4)
             TEXT (.1,.9)'escolha tipo linha'
        DEF END

        SYMBOL CHOICE (3,linha(i))

        ECHO DEF ecolin VIEW janela(3) TILL NEXT PROMPT
             SELECT (linha)
                  (1)  TEXT (.2,.4)'solida'
                  (2)  TEXT (.2,.4)'tracej'
                  (3)  TEXT (.2,.4)'pto-trac'
             FIN
        DEF END

        SYMBOL DESENHO (max,fig,npont,pi,delta,nfig,
                        pto(nfig,1))

        PICTURE DEF picture(nfig) VIEW janela(1)
             LINE pto(nfig,1) & linha(nfig) MAX
        DEF END

        nfig=nfig+1

END DIACELL
```

```
DIACELL DELECAO (max,janela,qtedel,npont,echo)

      IN     VIEWPORT   janela(4)          /* conjunto de viewports
             INTEGER    max                /* qte máxima de figuras
             ECHO       echo(max)          /* ecos das figuras
      TRANS  INTEGER    qtedel             /* qte figuras deletadas
                        npont              /* vetor de coordenadas
      PDECL  END

      LOC    PROMPT     del                /* prompt para deletar
             INTEGER    i                  /* contador de loop
      DDECL  END

      INIT
           PROMPT DEF del VIEW janela(4) TTLL NEXT ECHO
                TEXT (.1,.9)'selecione figura'
           DEF END

           i=1
      INIT END

      BODY

           SYMBOL PICK (nomefig)          /* seleciona figura

           ECHO DELETION (nomefig)        /* deleta repres visual

           UNTIL (nomefig=echo(i))
                i=i+1
           FIN

           npto(i)=-1                  /* cancela qte pto p/fig del
           qtedel=qtedel+1             /* ajusta qte de figuras del

END DIACELL
```

```
DIACELL DESENHO (max,tipo,pi,delta,indice,npont,pto)

     IN    INTEGER    max            /* qte máxima figuras
                      tipo           /* tipo de figura
                      indice         /* índice da fig no vetor ptos
           REAL       delta          /* dimensão da figura
           POINT      pi             /* ponto inicial
     OUT   POINT      pto(max,5)     /* vetor de coordenadas
           INTEGER    npont(max)     /* qte pontos por figura
     PDECL END

     BODY
          SELECT (tipo)
               (1)
                    pto(indice,1).x = pi.x
                    pto(indice,2).x = pto(indice,1).x + delta
                    pto(indice,3).x = pto(indice,1).x + delta/2
                    pto(indice,4).x = pto(indice,1).x

                    pto(indice,1).y = pi.y
                    pto(indice,2).y = pto(indice,1).y
                    pto(indice,3).y = pto(indice,1).y + delta
                    pto(indice,4).y = pto(indice,1).y
                    npont(indice) = 4
                    FIN
               (2)
                    pto(indice,1).x = pi.x
                    pto(indice,2).x = pto(indice,1).x + delta
                    pto(indice,3).x = pto(indice,2).x
                    pto(indice,4).x = pto(indice,1).x
                    pto(indice,5).x = pto(indice,1).x

                    pto(indice,1).y = pi.y
                    pto(indice,2).y = pto(indice,1).y
                    pto(indice,3).y = pto(indice,1).y + delta
                    pto(indice,4).y = pto(indice,3).y
                    pto(indice,5).y = pto(indice,1).y
                    npont(indice) = 5
                    FIN
               (3)
                    pto(indice,1).x = pi.x
                    pto(indice,2).x = pto(indice,1).x + delta/2
                    pto(indice,3).x = pto(indice,1).x + delta
                    pto(indice,4).x = pto(indice,2).x
                    pto(indice,5).x = pto(indice,1).x

                    pto(indice,1).y = pi.y
                    pto(indice,2).y = pto(indice,1).y - delta/2
                    pto(indice,3).y = pto(indice,1).y
                    pto(indice,4).y = pto(indice,2).y + delta
                    pto(indice,5).y = pto(indice,1).y
                    npont(indice) = 5
                    FIN
               FIN
END DIACELL
```

Na Sequência, são apresentadas listagens das rotinas em FORTRAN-77 correspondentes às Células de Diálogo :

```
: PROGRAMA PRINCIPAL DO EXEMPLO
      PROGRAM APLICACAO

      INTEGER          MAX,NFIG,QTEDEL,NPONT(10),LINHA(10)
      REAL             PTOX(10,5),PTOY(10,5)

      MAX=10

      CALL SISTEM (MAX,NFIG,QTEDEL,PTOX,PTOY,NPONT,LINHA)

      TYPE 1,NFIG
      FORMAT(' QTE FIGURAS CRIADAS=>',I)

      STOP
      END
```

```
      SUBROUTINE SISTEM (MAX,NFIG,QTEDEL,PTOX,PTOY,NPONT,LINHA)

      IMPLICIT NONE

      INTEGER          MAX,NFIG,QTEDEL,NPONT(MAX),LINHA(MAX)
      REAL             PTOX(MAX,5),PTOY(MAX,5)

      INTEGER          REG,PTO,MENCOM,MENFIG,MENLIN,COR,ECOP1
      INTEGER          ECOP2,ECOP3,OPFIM1,OPFIM2,PICTUR(10)
      INTEGER          IA(2),IB(2),IC(2),ID(2),IE(2),IF(2),IG(3),IH(3)
      INTEGER          II(4),IJ(3),IL(3),IM(3),IN(3),IO(3),IP(3)
      INTEGER          TIPFIG,LADO,TIPLIN,ECFIG1,ECFIG2,ECFIG3
      INTEGER          ECLIN1,ECLIN2,ECLIN3,DEL,EC
      REAL             RX(5),RY(5),JANELA(4,4)
      LOGICAL          T,F


      DATA IA/'CRIA','R   '/
      DATA IB/'DELE','TAR '/
      DATA IC/'ENCE','RRAR'/
      DATA ID/'SOLI','DA  '/
      DATA IE/'TRAC','EJ  '/
      DATA IF/'PTO-','TRAC'/
      DATA IG/'CRIA','Ç CO','NCL '/
      DATA IH/'DELE','Ç CO','NCL '/
      DATA II/'TIP ','FIG/','POS ','INIC'/
      DATA IJ/'CRIA','Ç TR','IANG'/
      DATA IL/'CRIA','Ç QU','AD  '/
      DATA IM/'CRIA','Ç LO','SANG'/
      DATA IN/'ENTR','E DI','MENS'/
      DATA IO/'ESC ','TIP ','LIN '/
      DATA IP/'SELE','Ç FI','GURA'/

      DATA T/.TRUE./
      DATA F/.FALSE./



      CALL GOPKS(5)              !ABERTURA GKS
      CALL GOPWK(4,8152,4)       !ABERTURA ESTACAO TRABALHO
      CALL GACWK(4)              !ATIVACAO ESTACAO TRABALHO

      RX(1)=0.
      RX(2)=1.
      RX(3)=1.
      RX(4)=0.
      RX(5)=0.

      RY(1)=0.
      RY(2)=0.
      RY(3)=1.
      RY(4)=1.
      RY(5)=0.
      CALL GPOLYL(5,RX,RY)

      RX(1)=.8
      RX(2)=.8
      RY(1)=0.
      RY(2)=1.
      CALL GPOLYL(2,RX,RY)
```

```
      RX(1)=.8
      RX(2)=1.
      RY(1)=.2
      RY(2)=.2
      CALL GPOLYL(2,RX,RY)

      RX(1)=.8
      RX(2)=1.
      RY(1)=.1
      RY(2)=.1
      CALL GPOLYL(2,RX,RY)


INCIALIZACAO DAS JANELAS

      JANELA(1,1) = 0.   !PARA TRABALHO
      JANELA(1,2) = 0.
      JANELA(1,3) = .8
      JANELA(1,4) = 1.

      JANELA(2,1) = .8   !PARA MENUS
      JANELA(2,2) = .2
      JANELA(2,3) = 1.
      JANELA(2,4) = 1.

      JANELA(3,1) = .8   !PARA ECOS
      JANELA(3,2) = .1
      JANELA(3,3) = 1.
      JANELA(3,4) = .2

      JANELA(4,1) = .8   !PARA PROMPTS
      JANELA(4,2) = 0.
      JANELA(4,3) = 1.
      JANELA(4,4) = .1


DEFINICAO DOS MENUS

      CALL GSVW(JANELA(2,1),JANELA(2,2),JANELA(2,3),JANELA(2,4)) !AREA MENUS

      MENCOM=1              !MENU DE COMANDOS
      CALL GCRSG(MENCOM)
      CALL GSVIS(MENCOM,F)
      CALL GSPCID(11)
      CALL GTX(.1,.8,8.IA)      !PICK=11
      CALL GSPCID(12)
      CALL GTX(.1,.6,8.IB)      !PICK=12
      CALL GSPCID(13)
      CALL GTX(.1,.4,8.IC)      !PICK=13
      CALL GCLSG

      MENFIG=2                !MENU DE FIGURAS
      CALL GCRSG(MENFIG)
      CALL GSVIS(MENFIG,F)
      CALL GSPCID(21)
      RX(1)=.1
      RX(2)=.5
      RX(3)=.3
      RX(4)=.1
```

```
      RY(1)=.7
      RY(2)=.7
      RY(3)=.8
      RY(4)=.7
      CALL GPOLYL(4,RX,RY)     !PICK=21
      CALL GSPCID(22)
      RX(1)=.1
      RX(2)=.5
      RX(3)=.5
      RX(4)=.1
      RX(5)=.1

      RY(1)=.5
      RY(2)=.5
      RY(3)=.6
      RY(4)=.6
      RY(5)=.5
      CALL GPOLYL(5,RX,RY)     !PICK=22
      CALL GSPCID(23)
      RX(1)=.1
      RX(2)=.3
      RX(3)=.5
      RX(4)=.3
      RX(5)=.1

      RY(1)=.3
      RY(2)=.2
      RY(3)=.3
      RY(4)=.4
      RY(5)=.3
      CALL GPOLYL(5,RX,RY)     !PICK=23
      CALL GCLSG


      MENLIN=3                 !MENU DE TIPOS DE LINHA
      CALL GCRSG(MENLIN)
      CALL GSVIS(MENLIN,F)
      CALL GSPCID(31)
      CALL GTX(.1,.8,8,ID)     !PICK=31
      CALL GSPCID(32)
      CALL GTX(.1,.6,8,IE)     !PICK=32
      CALL GSPCID(33)
      CALL GTX(.1,.4,8,IF)     !PICK=33
      CALL GCLSG


!INICIALIZACAO DE PROMPTS
      CALL GSVW(JANELA(4,1),JANELA(4,2),JANELA(4,3),JANELA(4,4)) !AREA PROMPT

      TIPFIG=4          !SOLICITA TIPO FIGURA E POS INICIAL
      CALL GCRSG(TIPFIG)
      CALL GSVIS(TIPFIG,F)
      CALL GTX(.1,.4,16,TI)
      CALL GCLSG

      LADO=5            !SOLICITA DIMENSAO
      CALL GCRSG(LADO)
      CALL GSVIS(LADO,F)
      CALL GTX(.1,.4,12,TN)
      CALL GCLSG
```

```
      TIPLIN=6          !SOLICITA TIPO LINHA
      CALL GCRSG(TIPLIN)
      CALL GSVIS(TIPLIN,F)
      CALL GTX(,1,,4,12,TO)
      CALL GCLSG

      DEL=7             !SOLICITA SELECAO FIGURA
      CALL GCRSG(DEL)
      CALL GSVIS(DEL,F)
      CALL GTX(,1,,4,12,TP)
      CALL GCLSG

INICIALIZACAO DOS ECOS

      CALL GSVW(JANELA(3,1),JANELA(3,2),JANELA(3,3),JANELA(3,4)) !AREA DE ECO

      ECOP1=8               !ECO DE OPERACAO 1
      CALL GCRSG(ECOP1)
      CALL GSVIS(ECOP1,F)
      CALL GTX(,1,,4,8,IA)
      CALL GCLSG

      ECOP2=9               !ECO DE OPERACAO 2
      CALL GCRSG(ECOP2)
      CALL GSVIS(ECOP2,F)
      CALL GTX(,1,,4,8,IB)
      CALL GCLSG

      ECOP3=10              !ECO DE OPERACAO 3
      CALL GCRSG(ECOP3)
      CALL GSVIS(ECOP3,F)
      CALL GTX(,1,,4,8,IC)
      CALL GCLSG

      OPFIM1=11             !ECO DE FINAL OPERACAO 1
      CALL GCRSG(OPFIM1)
      CALL GSVIS(OPFIM1,F)
      CALL GTX(,1,,4,12,IG)
      CALL GCLSG

      OPFIM2=12             !ECO FIM OPERACAO 2
      CALL GCRSG(OPFIM2)
      CALL GSVIS(OPFIM2,F)
      CALL GTX(,1,,4,12,IH)
      CALL GCLSG


INICIALIZACAO DOS ECOS
      CALL GSVW(JANELA(3,1),JANELA(3,2),JANELA(3,3),JANELA(3,4)) !AREA DE ECO

      ECFIG1=13         !ECO SELECAO FIGURA 1
      CALL GCRSG(ECFIG1)
      CALL GSVIS(ECFIG1,F)
      CALL GTX(,1,,4,12,IJ)
      CALL GCLSG

      ECFIG2=14         !ECO SELECAO FIGURA 2
      CALL GCRSG(ECFIG2)
      CALL GSVIS(ECFIG2,F)
      CALL GTX(,1,,4,12,IL)
      CALL GCLSG
```

```
      ECFIG3=15          !ECO SELECAO FIGURA 3
      CALL GCRSG(ECFIG3)
      CALL GSVIS(ECFIG3,F)
      CALL GTX(.1,.4,10,IM)
      CALL GCLSG

      ECLIN1=16          !ECO SELECAO LINHA 1
      CALL GCRSG(ECLIN1)
      CALL GSVIS(ECLIN1,F)
      CALL GTX(.2,.4,8.ID)
      CALL GCLSG

      ECLIN2=17          !ECO SELECAO LINHA 2
      CALL GCRSG(ECLIN2)
      CALL GSVIS(ECLIN2,F)
      CALL GTX(.2,.4,8.IE)
      CALL GCLSG

      ECLIN3=18          !ECO SELECAO LINHA 3
      CALL GCRSG(ECLIN3)
      CALL GSVIS(ECLIN3,F)
      CALL GTX(.2,.4,8.IE)
      CALL GCLSG


INICIALIZACAO DE VARIAVEIS
      QTEDEL = 0
      NFIG = 0
      COP = 0


CORPO
      DO WHILE (COP.NE.3.AND.NFIG.NE.MAX)

            IF(COP.NE.0)THEN
                  PC=(OPFIM1-1)*COP
                  CALL GSVIS(EC,F)
            END IF

            CALL GSVIS(MENCOM,T)
            CALL GROPC(4,1,SEL,PID) !SYMBOL CHOICE
            COP=PID - 10            !

            IF(COP.EQ.1)THEN
                  CALL GSVIS(ECOP1,T)  !ECO DE OPERACAO CRIACAO

                  CALL CRIAC(MAX,JANELA,MENCOM,MENFIG,MENLIN,
     1                  ECOP1,NFIG,NPONT,LINHA,PTOX,PTOY,PICTUR,
     2                  TIPFIG,LADO,TIPLIN,ECFIG1,ECFIG2,ECFIG3,
     3                  ECLIN1,ECLIN2,ECLIN3)

                  CALL GSVIS(OPFIM1,T) !ECO DE CRIACAO CONCLUIDA
            ELSE IF(COP.EQ.2)THEN
                  CALL GSVIS(ECOP2,T)  !ECO DE OPERACAO DELECAO
                  CALL DELEC(MAX,JANELA,QTEDEL,NPONT,PICTUR,DEL)
                  CALL GSVIS(ECOP2,F)
                  CALL GSVIS(OPFIM2,T) !ECO DE DELECAO CONCLUIDA
            ELSE
                  CALL GSVIS(ECOP3,T)  !ECO DE OPERACAO ENCERRAR
            END IF
```

```
END DO

CALL GECLKS      !FECHAMENTO GKS
RETURN
END
```

```
      SUBROUTINE CRIAC (MAX,JANELA,MENCOM,MENFIG,MENLIN,ECOP1,NFIG,
     1                  NPONT,LINHA,PTOX,PTOY,PICTUR,TIPFIG,LADO,
     2            TIPLIN,ECFIG1,ECFIG2,ECFIG3,ECLIN1,ECLIN2,ECLIN3)

      IMPLICIT NONE

      INTEGER           MAX,MENCOM,MENFIG,MENLIN,NFIG,NPONT(MAX),LINHA(MAX)
      INTEGER           PICTUR(1),ECOP1
      REAL              PTOX(MAX,5),PTOY(MAX,5),JANELA(4,4)

      INTEGER           SEG,PID,EC,N,INDICE,TIPFIG,LADO,TIPLIN,FIG
      INTEGER           ECFIG1,ECFIG2,ECFIG3,ECLIN1,ECLIN2,ECLIN3
      REAL              DELTA,PIX,PIY

      LOGICAL           T,F

      DATA              T/.TRUE./
      DATA              F/.FALSE./
      DATA              N/19/

CORPO
      INDICE=NFIG+1

      CALL GSVW(JANELA(1,1),JANELA(1,2),JANELA(1,3),JANELA(1,3)) !AREA TRABAL

      CALL GSVIS(MENCOM,F)    !MENU COMANDOS INVISIVEL
      CALL GSVIS(TIPFIG,T)    !PROMPT "TIPO FIG/POS INIC" VISIVEL
      CALL GSVIS(MENFIG,T)    !MENU FIGURAS VISIVEL

      CALL GRQPC(4,1,SEG,PID) !SYMBOL  CHOICE
      FIG=PID - 20            !

      CALL GSVIS(ECOP1,F)     !ECO DE OPERACAO CRIACAO

      EC=(ECFIG1-1)+FIG
      CALL GSVIS(EC,T)        !ECO TIPO FIGURA VISIVEL

      CALL GRQLC(4,1,PIX,PIY) !SYMBOL LOCATOR

      CALL GSVIS(TIPFIG,F)    !PROMPT "TIP FIG/POS IN" INVISIVEL
      CALL GSVIS(LADO,T)      !PROMPT "ENTRE DIMENS" VISIVEL

      CALL GRQVL(4,1,DELTA)   !SYMBOL VALUATOR

      CALL GSVIS(LADO,F)      !PROMPT "ENTRE DIMENS" INVISIVEL
      CALL GSVIS(MENFIG,F)    !MENU FIGURAS INVISIVEL
      CALL GSVIS(MENLIN,T)    !MENU TIPOS LINHA VISIVEL
      CALL GSVIS(TIPLIN,T)    !PROMPT "ESC TIP LIN" VISIVEL

      CALL GRQPC(4,1,SEG,PID) !SYMBOL CHOICE
      LINHA(INDICE)=PID - 40  !

      CALL GSVIS(EC,F)        !ECO TIPO FIGURA INVISIVEL

      EC=(ECLIN1-1)+LINHA(INDICE)
      CALL GSVIS(EC,T)        !ECO  TIPO LINHA VISIVEL

      CALL DFSEN(MAX,FIG,PIX,PIY,DELTA,INDICE,NPONT,PTOX,PTOY)

      NPONT(INDICE)=5
```

```
IF(PTG.EQ.1)NPONT(INDICE)=4

CALL GSPN(LINHA(INDICE))

CALL GSVW(JANELA(1,1),JANELA(1,2),JANELA(1,3),JANELA(1,4)) !AREA TRAB

CALL GCRSG(N)                                                    !CRIACAO
CALL GPOLYL(NPONT(INDICE),PTOX(INDICE,1),PTOY(INDICE,1))        !DA
CALL GCLSG                                                       !FIGURA
PICTUR(INDICE)=N                                                 !


N = N +1          !INCREMENTA QTDE SEGMENTOS GKS USADOS

NFIG=NFIG+1       !INCREMENTA QTDE FIGURAS CRIADAS

CALL GSVIS(EC,F)          !EÇO TIPO LINHA INVISIVEL
CALL GSVIS(MENLIN,F)      !MENU TIPOS LINHA INVISIVEL
CALL GSVIS(TIPLIN,F)
CALL GSVIS(EC,F)


RETURN
END
```

```
      SUBROUTINE DELEC (MAX,JANELA,QTEDEL,NPONT,PICTUR,DEL)

      IMPLICIT NONE

      INTEGER           MAX,QTEDEL,NPONT(MAX),PICTUR(MAX)
      REAL              JANELA(4,4)

      INTEGER           DEL,NOMFIG,I,PID
      LOGICAL           T,F



      DATA T/.TRUE./
      DATA F/.FALSE./

      CALL GSVIS(DEL,T)

EA TRABALHO
      CALL GSVW(JANELA(1,1),JANELA(1,2),JANELA(1,3),JANELA(1,4)) !AREA TRABAL

      CALL GROPC(4,1,NOMFIG,PID)  !SYMBOL PICK

      CALL GOLSG(NOMFIG)

      I=1
      DO WHILE (NOMFIG.NE.PICTUR(I))
              I=I+1
      END DO
      NPONT(I)=-1
      QTEDEL=QTEDEL+1

      CALL GSVIS(DEL,F)     !PROMPT INVISIVEL

      RETURN
      END
```

```
SUBROTINA QUE CALCULA OS PONTOS PARA AS FIGURAS

      SUBROUTINE DESEN(MAX,TIPO,PIX,PIY,DELTA,INDICE,PTOX,PTOY)

      INTEGER          MAX,TIPO,INDICE
      REAL             DELTA,PIX,PIY,PTOX(MAX,5),PTOY(MAX,5)


TRIANGULO
      IF(TIPO.EQ.1)THEN
         PTOX(INDICE,1)=PIX
         PTOX(INDICE,2)=PTOX(INDICE,1) + DELTA
         PTOX(INDICE,3)=PTOX(INDICE,1) + DELTA/2
         PTOX(INDICE,4)=PTOX(INDICE,1)

         PTOY(INDICE,1)=PIY
         PTOY(INDICE,2)=PTOY(INDICE,1)
         PTOY(INDICE,3)=PTOY(INDICE,1) + DELTA
         PTOY(INDICE,4)=PTOY(INDICE,1)


QUADRADO
      ELSE IF(TIPO.EQ.2)THEN
         PTOX(INDICE,1)=PIX
         PTOX(INDICE,2)=PTOX(INDICE,1) + DELTA
         PTOX(INDICE,3)=PTOX(INDICE,2)
         PTOX(INDICE,4)=PTOX(INDICE,1)
         PTOX(INDICE,5)=PTOX(INDICE,1)

         PTOY(INDICE,1)=PIY
         PTOY(INDICE,2)=PTOY(INDICE,1)
         PTOY(INDICE,3)=PTOY(INDICE,1) + DELTA
         PTOY(INDICE,4)=PTOY(INDICE,3)
         PTOY(INDICE,5)=PTOY(INDICE,1)


LOSANGO
      ELSE IF(TIPO.EQ.3)THEN
         PTOX(INDICE,1)=PIX
         PTOX(INDICE,2)=PTOX(INDICE,1) + DELTA/2
         PTOX(INDICE,3)=PTOX(INDICE,1) + DELTA
         PTOX(INDICE,4)=PTOX(INDICE,2)
         PTOX(INDICE,5)=PTOX(INDICE,1)

         PTOY(INDICE,1)=PIY
         PTOY(INDICE,2)=PTOY(INDICE,1) - DELTA/2
         PTOY(INDICE,3)=PTOY(INDICE,1)
         PTOY(INDICE,4)=PTOY(INDICE,2) + DELTA
         PTOY(INDICE,5)=PTOY(INDICE,1)

      END IF

FIM
      RETURN
      END
```