INTERFACE GRAFICA PARA O DESENVOLVIMENTO DE ESPECIFICAÇÕES EM LOTOS

Maria Teresa Silva de Moura
Paulo Roberto Freire Cunha

Departamento de Informática
Universidade Federal de Pernambuco
50739, Recife, PE

RESUMO

Este trabalho apresenta uma ferramenta para o desenvolvimento de especificações em LOTOS (Language Of Temporal Ordering Specification), uma técnica de descrição formal desenvolvida para a especificação de protocolos e sistemas distribuídos em geral. A interface gráfica foi definida com o intuito de tornar o processo de desenvolvimento de especificações mais amigável e é baseada numa metodologia de desenvolvimento estruturado, que integra as representações gráfica e textual.

ABSTRACT

This work presents a tool for the development of LOTOS (Language Of Temporal Ordering Specification) specifications, a formal description technique developed for the specification of protocols and distributed systems in general. The graphical interface provides a more friendly specification development process and is based on a structured development methodology, which integrates graphical and textual representations.

1. INTRODUÇÃO

As técnicas de descrição formal (TDFs) foram desenvolvidas com o intuito de fornecer meios para o desenvolvimento de descrições ou especificações formais de sistemas distribuídos de uma maneira clara, precisa e sem ambiguidades. A linguagem LOTOS ([Brink 86], [BoBri 87], [ISO 88]) é uma técnica de descrição formal para a especificação de sistemas distribuídos, mais precisamente protocolos e serviços do Modelo OSI. Uma especificação em LOTOS possui um componente estático para a definição de estruturas de dados, que é baseado na álgebra para especificação de tipos abstratos de dados ACT ONE [EhMha 85] e um componente dinâmico para a definição do comportamento da especificação, que é uma extensão de CCS (Calculus of Communicating Systems) [Milne 80]. A linguagem LOTOS é baseada num

modelo matemático formal com sintaxe e semântica bem definidas e vem sendo utilizada com sucesso na especificação de protocolos e serviços do Modelo OSI. Em particular, este trabalho destaca a experiência no uso desta linguagem para a especificação de sistemas distribuídos de um modo geral.

A utilização de representações gráficas para a especificação de sistemas distribuídos é uma ferramenta poderosa visto que atribui clareza e oferece uma interface humana amigável. No entanto, é importante observar que à medida que informações mais detalhadas e estruturas mais complexas são expressas graficamente, podem ser comprometidos o entendimento e clareza do gráfico. Assim, acreditamos que uma boa interação entre o gráfico e o texto ofereça a melhor opção para o desenvolvimento de uma especificação clara e precisa. A utilização de diagramas como suporte às descrições textuais já é uma prática utilizada, no entanto, na sua maioria estes diagramas eram produzidos manualmente e utilizados informalmente. Alguns trabalhos nesta área propõem a definição de ambientes de desenvolvimento utilizando diagramas em conjunto com as descrições textuais ([DeSch 86], [KrMNg 89]). Neste trabalho, apresentamos uma interface gráfica como ferramenta de auxílio ao processo de construção de especificações em LOTOS, baseada numa metodologia para o desenvolvimento estruturado de especificações, que integra o ferramental gráfico e a representação textual atribuindo, desta forma, clareza e precisão, requisitos básicos quando se trata de especificação de sistemas distribuídos.

O restante do trabalho está estruturado da seguinte forma. Na seção 2 apresentaremos a linguagem LOTOS acompanhada da nossa representação gráfica para seus principais construtores; na seção 3 discutiremos a metodologia para o desenvolvimento estruturado de especificações e em seguida apresentaremos a interface gráfica; na seção 4 ilustraremos a utilização da interface através de um exemplo; e finalmente, na seção 5, concluiremos o trabalho.

2. LOTOS E A REPRESENTAÇÃO GRÁFICA DOS PRINCIPAIS OPERADORES

Nesta seção, apresentaremos os principais aspectos da parte dinâmica de LOTOS, a qual permite a definição do comportamento observável da especificação, e que será utilizada no ambiente de desenvolvimento de especificações, uma vez que, através deste, pretendemos permitir a definição do comportamento geral da especificação.

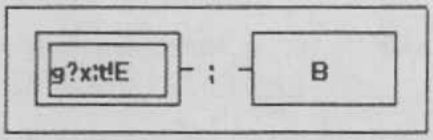
O comportamento observável de uma especificação em LOTOS é descrito através de processos que se comunicam entre si e o ambiente externo através dos pontos de interação, sendo esta interação representada por um evento, unidade de comunicação sincronizada [BoBri 87]. Desta forma, cada processo LOTOS é visto como uma caixa preta onde somente os pontos de interação ou portas são conhecidos externamente. Uma interação entre processos LOTOS ocorre quando os processos envolvidos habilitam o

mesmo evento. Uma interação pode envolver ou não passagem de valor. Se a interação envolve passagem de valor, a oferta de evento se dá através da associação de valores ou variáveis aos pontos de interação. Assim, podemos dizer que **g?x:t** indica que o processo pode interagir através do ponto de interação **g** para aceitar um valor **x** do tipo **t**, e **g!E** indica que o processo pode interagir através da porta **g** oferecendo o valor **E**.

O comportamento de um processo LOTOS é descrito através de expressões de comportamento. Uma expressão de comportamento é formada a partir de interações e operações. A seguir apresentamos os principais construtores de LOTOS juntamente com a representação gráfica para cada um deles, definida em [MoCun 89], a qual será utilizada na construção dos diagramas da especificação no ambiente de desenvolvimento que será apresentado a seguir. Decidimos por definir uma nova representação gráfica para LOTOS porque não pretendemos mapear todos os construtores da linguagem no gráfico, mas apenas aqueles necessários à definição da estrutura geral da especificação, sem levar em consideração informações mais detalhadas. A nossa preocupação é de prover uma interface gráfica simples e de fácil adequação ao nosso ambiente de desenvolvimento estruturado de especificações em LOTOS.

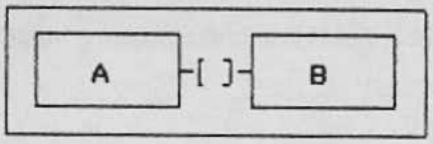
i) **ação**: indica sequencialidade entre uma oferta de evento e uma expressão de comportamento. A expressão **a;B** indica que a expressão de comportamento B pode ser executada após a ocorrência do evento a. Uma oferta de eventos que precede a ativação de um processo é representada por um retângulo em linha dupla, sendo que o interior deste retângulo deverá conter o nome do ponto de interação seguido dos atributos (valores ou variáveis), enquanto que um processo é descrito por uma retângulo em linha simples que contem no seu interior o nome do processo e os seus parâmetros, caso existam. Os pontos de sincronização do processo são representados por arcos, os quais podem ser colocados em qualquer um dos quatro lados do retângulo, de forma que o nome do evento é etiquetado próximo ao arco. Uma operação de sequenciamento entre a oferta de eventos g?x:t!E e o processo B é representada da seguinte forma:

g?x:t!E; B



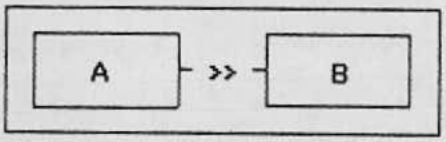
ii) **escolha**: indica uma escolha não determinística entre expressões de comportamento de acordo com o evento oferecido pelo ambiente. Uma operação de escolha entre os processo A e B é representada da seguinte forma:

A [ ] B



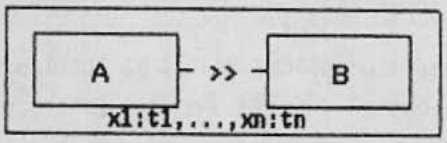
iii) composição sequencial: expressa a sequencialidade entre processos, onde a terminação com sucesso de um processo habilita a execução do processo seguinte. A composição sequencial entre os processos A e B e representada da seguinte forma:

A >> B



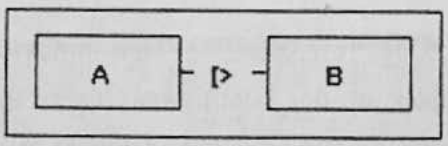
Esta operação pode ser generalizada para permitir a passagem de informações entre os processos, da seguinte forma:

A >> accept $x_1:t_1$, ... ,$x_n:t_n$ in B



iv) desabilitação: expressa a situação onde um processo pode interromper a execução de outro. Esta desabilitação acontece se o evento inicial do processo desabilitador ocorrer antes da terminação com sucesso do outro processo. Um processo A que pode ser interrompido por um processo B é representado da seguinte forma:

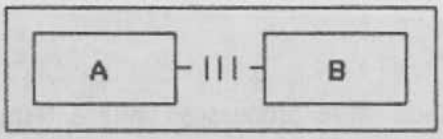
A [> B



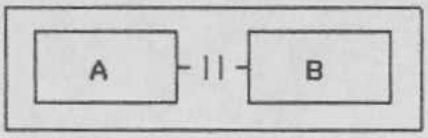
v) composição paralela: expressa a execução concorrente de processos. Em LOTOS existem três tipos de paralelismos: composição paralela não sincronizada, onde os processos executam em paralelo mas não sincronizam, ou seja, os processos não se comunicam entre si; composição paralela com plena sincronização, onde os processos executam em paralelo com relação a todos os eventos; e finalmente a

composição paralela geral, onde os processos executam em paralelo e sincronizam com relação a uma lista de eventos para sincronização. Estes três tipos de paralelismos são representados como segue:

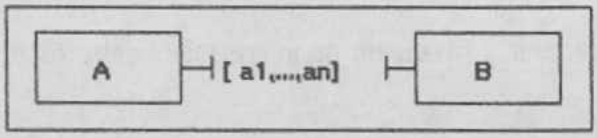
- sem sincronização: **A ||| B**



- com plena sincronização: **A || B**

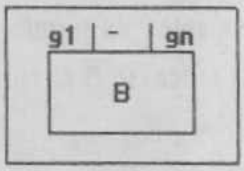


- caso geral: **A |[a_1, ... ,a_n]| B**



vi) **operador "hide"**: no caso de utilização da composição paralela, pode-se ter a presença de eventos que serão usados apenas para a comunicação interna entre os processos. Estes eventos não devem portanto ser acessíveis pelo ambiente. A situação em que os eventos g_1,...,g_n do processo B são escondidos do observador externo é representada da seguinte forma:

hide g_1, ... ,g_n in B



Neste caso, como na representação gráfica dos eventos observáveis de um processo descrita anteriormente, os arcos podem ser colocados em qualquer um dos quatro lados do retângulo. Uma consideração importante a fazer é o fato de que todas as representações gráficas definidas valem tanto da direita para a esquerda, conforme foram apresentadas, quanto de cima para baixo, ou seja, o diagrama pode ser construído tanto na horizontal quanto na vertical.

3. APRESENTAÇÃO DA INTERFACE

Conforme dissemos anteriormente, nossa ferramenta para a construção de especificações é baseada numa metodologia para o desenvolvimento estruturado de especificações em LOTOS [Moura 89], a qual estende os princípios bem fundamentados de modularidade [Myers 78], abstrações, desenvolvimento *top-down*, e de programação orientada a mensagens [Cunha 81], e utiliza diagramas para representar graficamente a configuração do sistema como ferramenta de auxílio ao desenvolvimento e documentação da especificação.

A metodologia define que o desenvolvimento da especificação começa com a decomposição do sistema em módulos principais ou processos LOTOS. Em seguida deve ser descrito o fluxo de informações entre os processos, que, no caso de LOTOS, significa a definição dos pontos de sincronização e dos respectivos valores associados a cada ponto. A partir daí, deve ser definido o tipo de associação entre os processos, ou seja, identificar o relacionamento entre os processos durante a execução, procurando estruturar a solução do problema de modo a atingir o maior grau de paralelismo entre os processos. Uma vez que a primeira decomposição está completa, deve ser efetuada uma validação desta com os requisitos iniciais do sistema. Desta forma, estas informações devem ser representadas graficamente, utilizando a representação gráfica para LOTOS. Em seguida, deve ser definida uma representação textual para o diagrama estruturado na fase anterior. A especificação continua com a decomposição do sistema através de refinamentos sucessivos, tal que os principais aspectos da especificação vão sendo adicionados de forma que o sistema é descrito sistematicamente através de uma estrutura hierárquica de módulos independentes. Finalmente, a especificação deve ser concluída através da definição dos processos do nível mais baixo da hierarquia. Uma descrição mais detalhada da metodologia se encontra em [Moura 89].

Desta forma, tomando como base a metodologia, procuramos estruturar a interface tal que esta propicia o desenvolvimeto modular da especificação, onde novas características vão sendo adicionadas através de refinamentos sucessivos nos módulos do sistema. A interface gráfica, dentro desta metodologia, fornece, a partir das informações do sistema sendo especificado, a geração automática de diagramas e expressões de comportamento LOTOS. A cada refinamento a interface vai construindo as representações gráfica e textual do sistema de forma que no final temos a representação gráfica completa dos principais aspectos do sistema e a especificação LOTOS completa do comportamento geral da especificação.

As informações sobre a especificação são armazenadas de forma que a interface mantem uma base de dados para cada sistema especificado. Assim, para a definição das representações gráfica e textual do sistema, a interface analisa a base de dados e gera como saída o diagrama e as expressões de comportamento LOTOS para a especificação. A figura 1 apresenta o esquema geral de funcionamento da interface, onde destacamos o fluxo de dados para a interação entre a interface e o usuário.

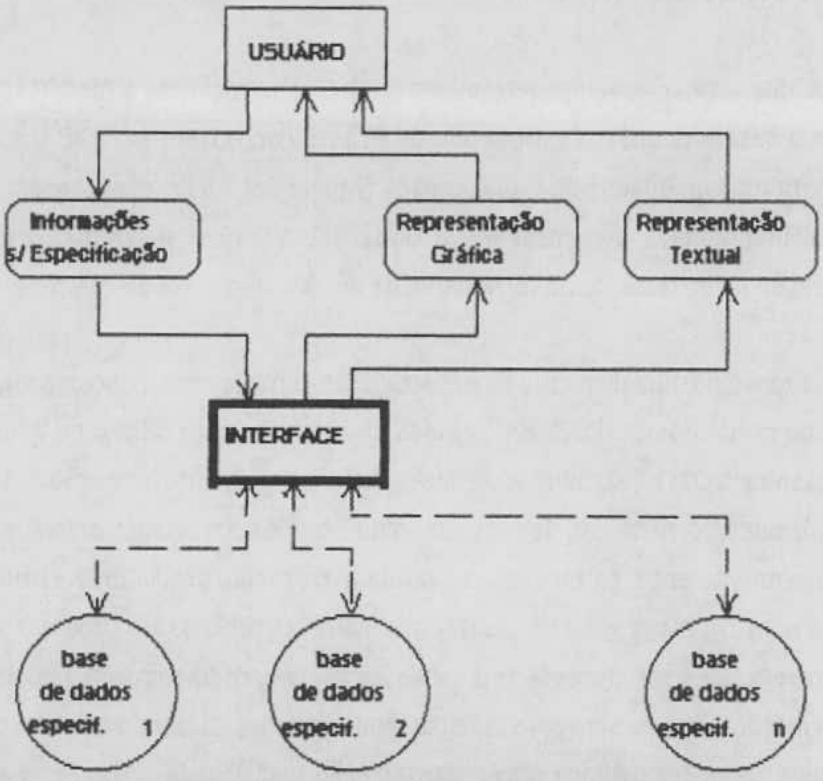
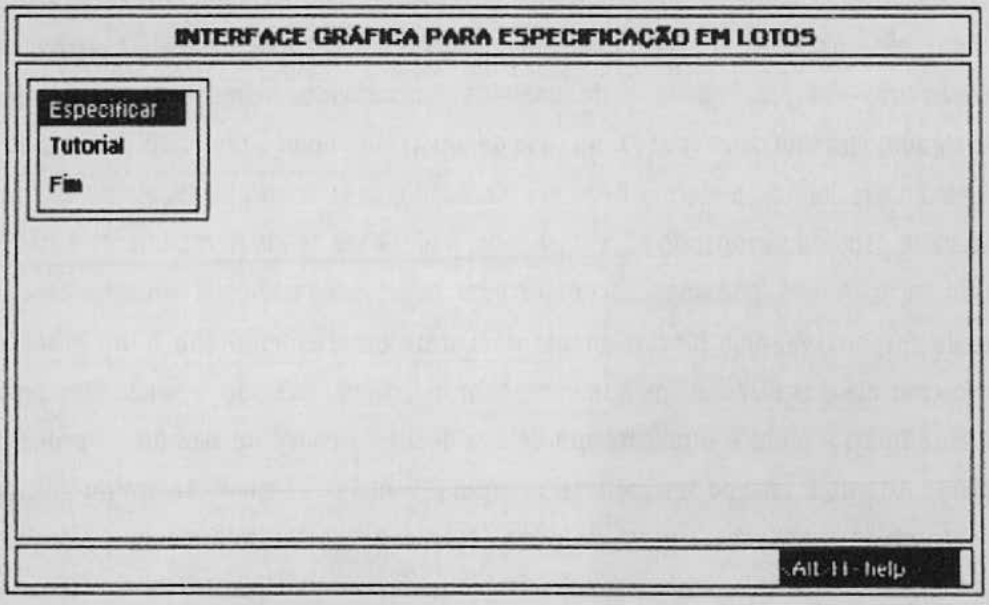


Fig 1: ESQUEMA GERAL DE FUNCIONAMENTO DA INTERFACE

A interação do usuário com a interface se dá através de janelas. Existem três tipos de janelas, **janela de opções**, **janela de diálogo** e **janela de informação**. A janela de opções é caracterizada por apresentar um menu de opções onde o usuário pode selecionar a opção desejada. A movimentação no menu é indicada através do deslocamento de uma barra em vídeo reverso, de forma que uma vez que o usuário esteja posicionado na opção desejada, a tecla <Enter> confirma a escolha. A janela de diálogo é utilizada pela interface para obter informações do usuário sobre o sistema sendo especificado. A janela de informação é utilizada pela interface para apresentar as representações gráfica e textual do sistema, quando requeridas pelo usuário e para apresentar outras informações, as quais nos referenciaremos a seguir. Durante a interação é possível, em qualquer situação, pedir ajuda (*help*) ao sistema, que é fornecida através de uma explicação sucinta da opção em destaque, caso o usuário esteja posicionado num menu de opções, ou do tipo de informação requerida, caso o usuário esteja posicionado numa janela de diálogo fornecendo informações sobre o sistema sendo especificado.

A primeira tela da interface, ilustrada na figura abaixo, apresenta uma janela de opções, onde é possível selecionar a opção **Especificar** para o desenvolvimento de especificações, utilizar a opção **Tutorial** ou finalizar a sessão através da opção **Fim**. Em particular, a opção tutorial apresenta-se como uma ferramenta

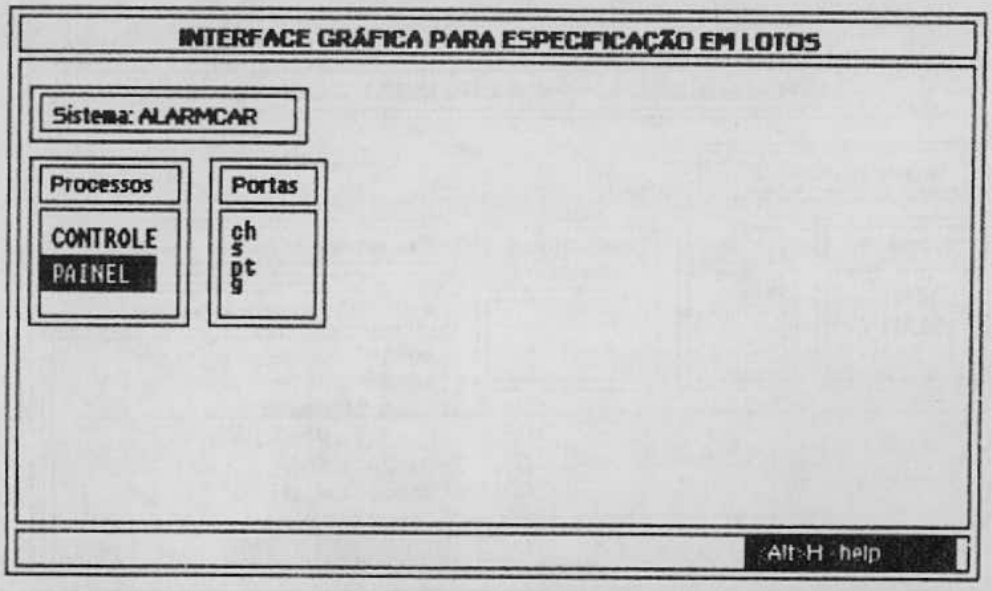
auxiliar com o intuito de oferecer ao usuário informações de como utilizar a interface bem como suas principais características.



De acordo com esta ferramenta é possível definir novas especificações ou alterar/visualizar especificações de sistemas já definidas e armazenadas na base de dados correspondente. No caso de inclusão de uma nova especificação, a interface pede que seja informado o nome do sistema que vai ser especificado. No caso de sistemas já definidos a interface apresenta uma janela com o nome de todos os sistemas especificados e gravados em disco. O usuário deve selecionar o sistema desejado. Ao ser escolhido o sistema a interface apresenta as informações sobre o primeiro refinamento e a partir daí é possível alterar as informações ou simplesmente visualizar as representações gráfica e textual da especificação.

A partir da definição do nome do sistema, as informações sobre o mesmo são requeridas através de janelas de diálogo e janelas de opções. A primeira janela (PROCESSOS) pede que seja informado o nome dos processos que decompõem o sistema; a segunda janela (PORTAS) pede o nome das portas associadas a cada processo; a terceira janela (PARÂMETROS) pede o nome e tipo dos parâmetros de cada processo, caso existam. O processo cujas portas ou parâmetros estão sendo informados fica destacado através de uma barra em vídeo-reverso. A quarta janela (FUNCIONALIDADE) pede que seja informada a funcionalidade associada ao sistema, caso se trate da primeira decomposição, ou do processo sendo refinado, caso contrário. Finalmente, a quinta janela (ASSOCIAÇÃO ENTRE OS PROCESSOS) é uma janela de opções que apresenta um menu dos operadores LOTOS para que seja definido o tipo de relacionamento entre os processos durante a execução.

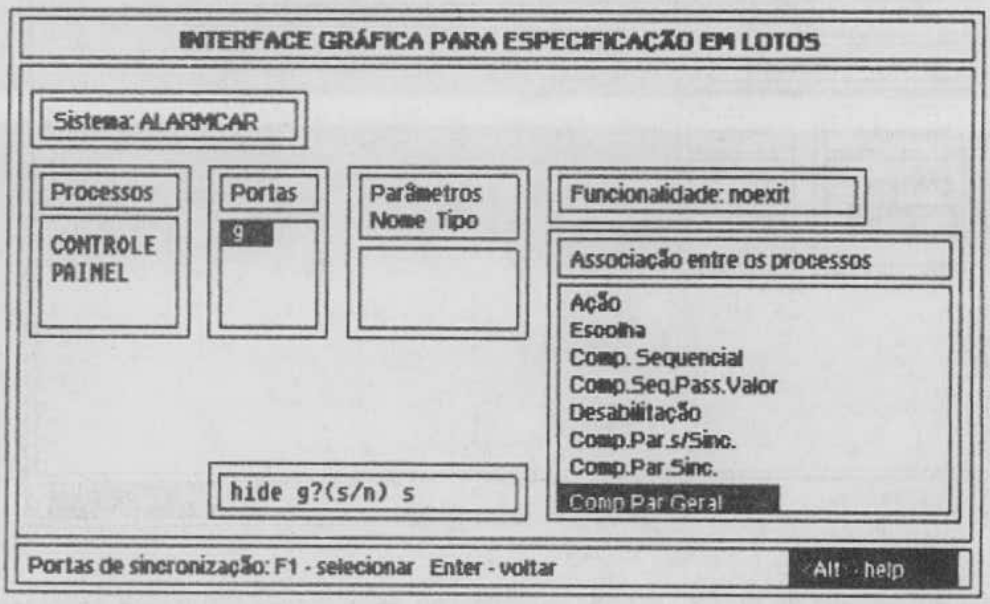
Para ilustrar esta interação, tomemos como exemplo um sistema de alarme de automóvel, o qual controla o funcionamento do carro através de um sinalizador de alarme e do mecanismo de abrir e fechar a porta do veículo. A alarme funciona da seguinte forma: se o carro estiver desligado e alguém entra no carro (abre/fecha a porta), não irá conseguir ligar o carro até que sinalize o alarme uma vez. O alarme deverá ser sinalizado uma segunda vez para que o carro continue funcionando. Uma vez que o carro estiver funcionando, se alguém entra no carro (por ex. no caso de um assalto onde o motorista é obrigado a sair do carro, dando lugar ao assaltante), o alarme deve ser sinalizado, caso contrário depois de algum tempo o carro pára. Se o carro estiver funcionando e for desligado, poderá ser ligado normalmente a não ser que o motorista saia do carro. Assim, podemos decompor este sistema inicialmente em dois processos, um processo Controle responsável pelo funcionamento do alarme propriamente dito e um processo Painel responsável pelo controle dos dispositivos que compõem o sistema, que são o sinalizador de alarme, o mecanismo de abre/fecha a porta e o mecanismo de liga/desliga a chave de ignição. O processo Painel recebe do ambiente externo o sinal do sinalizador de alarme (evento **s**), os sinais de liga ou desliga a chave de ignição (evento **ch**) e abre/fecha a porta do carro (evento **pt**); estas informações são passadas ao processo Controle. A fim de simular uma temporização para que possamos expressar a situação em que o sistema espera um determinado tempo t para que o alarme seja sinalizado, adicionaremos um fluxo de informação entre o processo Controle e o ambiente externo de forma que o processo Controle passe um sinal para o ambiente para a ativação de um contador de temporização (evento **t**). O ambiente externo passa, por sua vez, um sinal para parada do carro, em caso de estouro da temporização. Os dois processos podem ser colocados em paralelo e sincronizam a sinalização do alarme, e os sinais de liga ou desliga a chave de ignição e abre/fecha a porta do carro através de um evento **g**, o qual pode ser escondido do ambiente. A figura abaixo mostra um exemplo de interação para definir este sistema.



Com relação à janela de opções para escolha do tipo de associação entre os processos durante a exucução é importante ressaltar que:

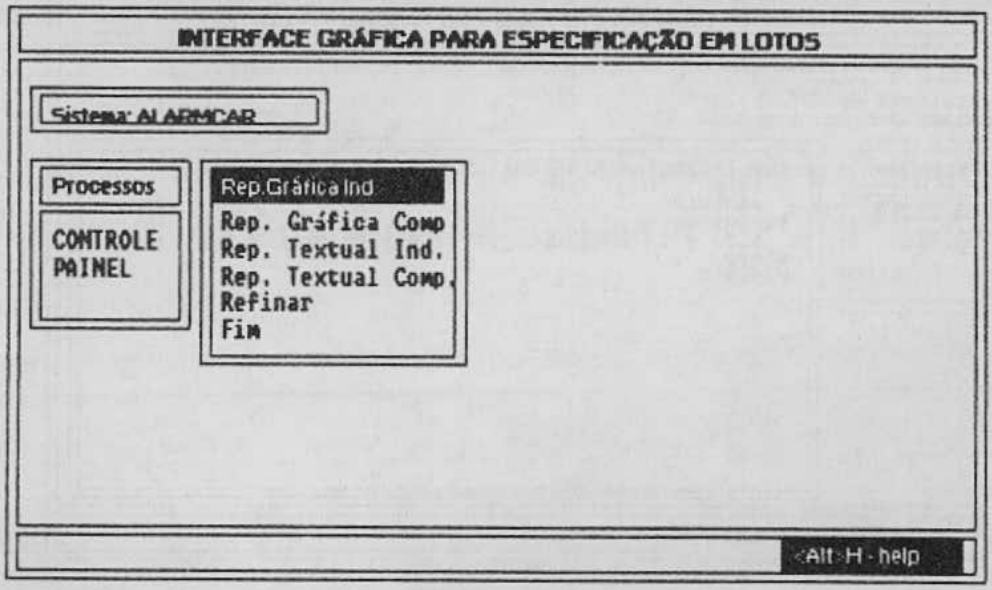
- caso seja fornecido apenas um nome de processo na primeira janela (PROCESSOS), a interface assume a operação de AÇÃO na quinta janela (ASSOCIAÇÃO ENTRE OS PROCESSOS), já que somente esta envolve apenas um processo, e pede que seja informada a oferta de eventos que precede a ativação do referido processo;
- se a operação escolhida for a COMPOSIÇÃO SEQUENCIAL COM PASSAGEM DE VALOR, a terceira janela (PARÂMETROS) solicita as variáveis a serem passadas entre cada dois processos, de forma que os dois processos cujos parâmetros para passagem estão sendo informados ficam destacados através de uma barra em vídeo-reverso.
- se a operação escolhida for a COMPOSIÇÃO PARALELA COM PLENA SINCRONIZAÇÃO, a segunda janela (PORTAS) torna-se uma janela de opções apresentando a relação das portas para que sejam informados os eventos não observáveis durante a sincronização ou seja, as portas *hide*.
- caso a operação desejada for a COMPOSIÇÃO PARALELA GERAL, a segunda janela (PORTAS) torna-se uma janela de opções apresentando a lista de portas em comum, onde deverão ser selecionadas as portas de sincronização. A cada porta selecionada, uma janela de diálogo pergunta se a referida porta é uma porta não observável externamente.

No caso do exemplo do Alarme, como pode ser visualizado na figura abaixo, após ser escolhida a operação de composição paralela geral, a segunda janela (PORTAS) apresenta a porta em comum entre os processos Controle e Painel. Como esta porta de sincronização não é observável externamente, o usuário responde sim à pergunta na janela de diálogo que indaga se a referida porta deve ser escondida.

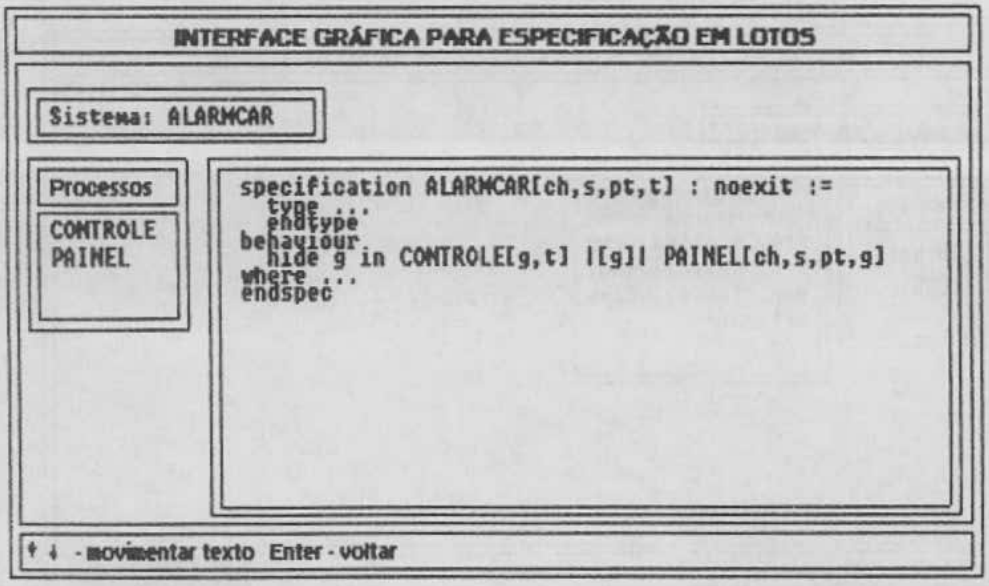


As opções de visualização das representações gráfica e textual são oferecidas numa janela de opções sempre que uma nova decomposição sobre algum processo é efetuada ou selecionada, como mostra a figura a seguir. É apresentada também a opção **REFINAR**, onde é permitido escolher qualquer um dos processos informados em fases anteriores e efetuar sua decomposição caso o processo ainda não tenha sido refinado; caso contrário, as informações sobre o refinamento são apresentadas e o usuário pode efetuar alguma alteração ou simplesmente confirmar o refinamento (utilizando a tecla <Esc>) e selecionar as opções de visualização desta parte da especificação ou o todo. É através da opção refinar que o usuário pode selecionar uma determinada parte da especificação. Durante todo o desenvolvimento da especificação, uma janela de informação, situada no canto superior esquerdo da área central da tela informa o nome do sistema e o nome do processo sendo refinado, juntamente com seus pontos de interação e parâmetros, caso se trate de um refinamento.

Com relação à opção de refinamentos, quando selecionada, transforma a janela PROCESSOS numa janela de opções onde o usuário pode selecionar o processo que deseja refinar. A lista de processos para refinar é apresentada de forma modular, de acordo com os refinamentos efetuados no sistema. Caso o processo que se deseja refinar não esteja presente na lista de processos, a tecla F1 dá acesso a uma nova lista de processos.



As figuras a seguir mostram as representações gráfica e textual da primeira decomposição do exemplo do relógio.



Um aspecto fundamental quando se trata de desenvolvimento modular está relacionado com a manutenção da validade das informações ou consistência dos dados [YuNak 85]. Em se tratando de desenvolvimento modular, alterações na estrutura interna de um determinado módulo não se propagam pelo resto do sistema; em contrapartida, qualquer modificação na interface de um módulo pode levar a inconsistências no resto do sistema. A deteção e correção destas inconsistências não deve ser tarefa apenas do projetista, devido ao fato de que nem sempre é simples detectar tais inconsistências. Ademais, o acúmulo de inconsistências na base de dados pode levar à total desorganização das informações contidas e a uma maior dificuldade no estabelecimento da consistência dos dados. Neste sentido, acreditamos que o ambiente de desenvolvimento deve ser responsável pela manutenção da integridade dos dados, verificando com rapidez os efeitos de cada alteração e corrigindo, se possível, os módulos afetados, de forma que o projetista fica protegido da ocorrência de tais inconsistências. A seguir, apresentaremos os aspectos principais considerados no nosso ambiente de desenvolvimento com relação à consistência das informações quando descreveremos os procedimentos realizados pela interface para consistência dos dados sem intervenção do projetista e as mensagens de erro apresentadas pela interface induzindo o projetista a efetuar correção nos dados.

No caso de alteração em especificações já definidas é possível efetuar modificação em qualquer uma das informações de cada refinamento, inclusive a remoção de processos. Neste caso, para manter a consistência das informações, a interface analisa a base de dados para a retirada do refinamento deste processo, caso exista, e de todos os processos que tiverem sido criados a partir deste refinamento. É possível efetuar também a retirada de portas de um determinado processo; neste caso, a interface também analisa e altera a base de dados com a finalidade de manter a estrutura do refinamento.

Quando o usuário está efetuando uma decomposição em um determinado processo é importante observar que não é possível acrescentar novas portas externas nos subprocessos, a não ser que sejam portas não observáveis, definidas apenas para a sincronização interna entre os subprocessos. Caso a estrutura do refinamento seja violada, a interface apresenta a seguinte mensagem na área reservada para as mensagens de interação com o usuário:

ERRO! Não é permitida a definição de novas portas externas

e fica posicionada na janela PORTAS para que o usuário possa efetuar a correção nos dados apresentados. É importante observar que, conforme dissemos anteriormente, uma janela de informação situada no canto superior esquerdo da área central da tela apresenta o nome do processo sendo refinado e suas portas associadas, além de outras informações, o que informa ao usuário sobre quais portas podem ser utilizadas na definição do refinamento.

Com relação ao operador de ação que representa a sequencialidade entre uma oferta de eventos e um processo é necessário que a oferta de eventos que precede a ativação do processo informado na janela PROCESSOS envolva apenas as portas pertencentes ao processo sendo refinado. Caso isto não ocorra, a interface apresenta a mensagem

ERRO! A oferta de eventos no processo refinado contem portas não definidas

e espera a confirmação da referida informação. O usuário pode visualizar as portas pertencentes ao processo sendo refinado conforme ressaltamos no caso anterior.

O operador de composição paralela com plena sincronização exige que as portas definidas sejam comuns a todos os processos, isto porque este tipo de paralelismo efetua sincronização com relação a todos o eventos. Caso esta regra seja violada a interface apresenta a mensagem

ERRO! Você tem portas diferentes nos diversos processos

e posiciona-se na janela de diálogo para que o usuário altere o nome das portas associadas a cada processo ou persista com as mesmas informações e altere o operador de associação entre os processos na janela de opções para escolha do relacionamento entre os processos.

A geração do diagrama é feita a partir de análise à base de dados. Com a finalidade de possibilitar a construção de diagramas próximos da estrutura ideal, com o *lay-out* idealizado pelo usuário, definimos um algorítmo para a geração da representação gráfica da especificação. O diagrama é montado alternando-se a direção em cada refinamento, isto é, se a primeira decomposição do sistema é representada no gráfico horizontalmente, os refinamentos nos processos que decompõem o sistema são desenhados na vertical, os refinamentos em cada um destes novos processos são desenhados na horizontal e assim por diante, de modo que, desta forma, o diagrama cresce proporcionalmente nas duas direções. Embora este algorítmo não seja sofisticado o bastante para produzir o *lay-out* ótimo, ele é rápido apresentado-se como uma maneira eficiente de gerar a representação gráfica a qual poderá ser utilizada como ponto de partida para

futuros melhoramentos no *lay-out* do diagrama através de edição manual. Este algorítmo tem sido utilizado e vem apresentando bons resultados. Além do mais, ao ser apresentada a representação gráfica a interface dá ao usuário a opção de redesenhar o diagrama. Este procedimento funciona da seguinte forma: para iniciar a geração do gráfico é feita uma escolha aleatória da direção dos processos que compõem o primeiro refinamento de forma que se o usuário seleciona a opção de redesenhar o gráfico, o diagrama é montado a partir da direção oposta à anterior. Isto faz com que, em alguns casos, um diagrama que não esteja bem posicionado na tela possa ser melhor visualizado.

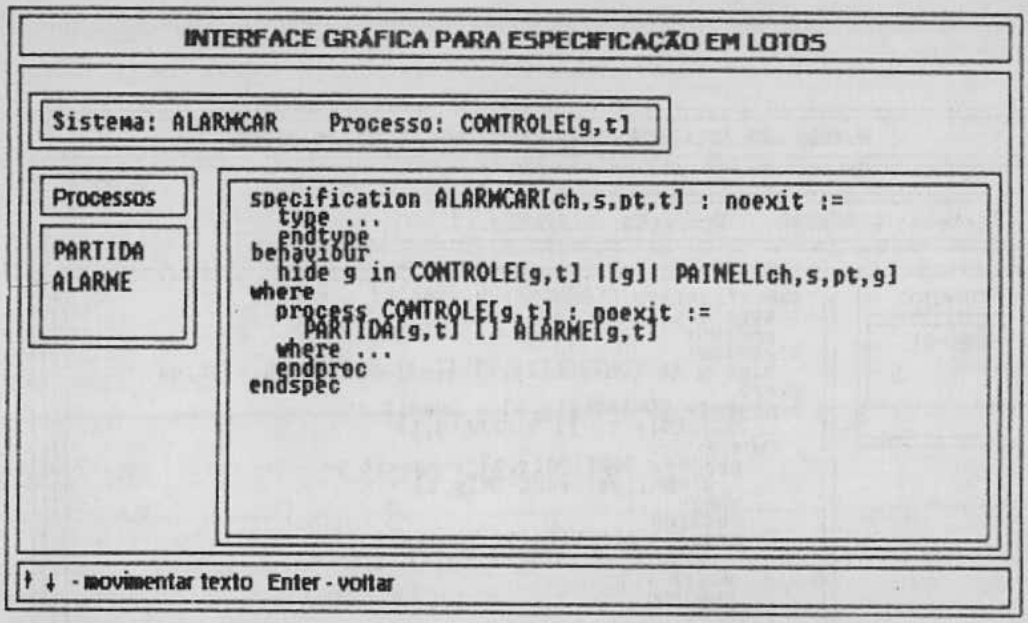
Embora tenhamos definido a opção de visualizar determinadas partes da especificação e o algorítmo apresentado para possibilitar um melhor posicionamento do gráfico na tela, é óbvio que ocorrerão situações em que se deseja visualizar toda a representação gráfica na tela e esta será suficientemente grande para impossibilitar um posicionamento completo. Com a finalidade de contornar este problema, a interface apresenta o recurso de rolagem de linha/coluna e de tela (*scroll*) para que o usuário possa, na medida do possível, visualizar de forma eficiente a representação gráfica da especificação. Esta rolagem de tela pode ser efetuada na horizontal ou na vertical utilizando as teclas de direção, com avanço pequeno (uma coluna para direita ou para esquerda e uma linha para cima ou para baixo), e avanço maior (10 colunas para direita ou para a esquerda e 10 linhas para cima ou para baixo).

Nesta seção apresentamos o funcionamento e alguns dos recursos oferecidos pela interface. Um exemplo passo-a-passo para a definição do sistema de alarme pode ser encontrado na próxima seção.

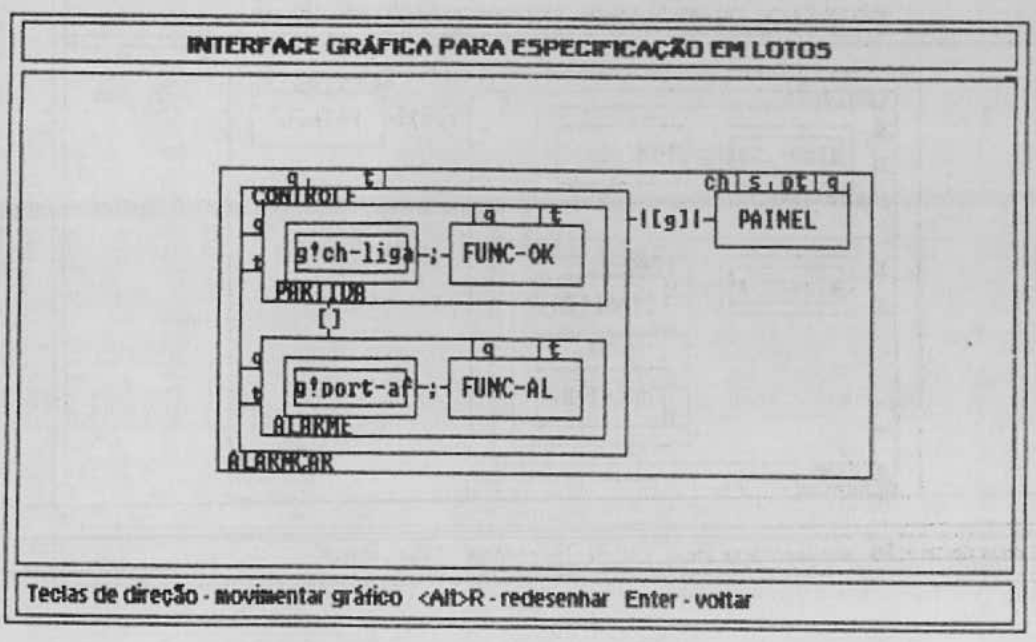
4. DESENVOLVIMENTO ESTRUTURADO DE ESPECIFICAÇÕES

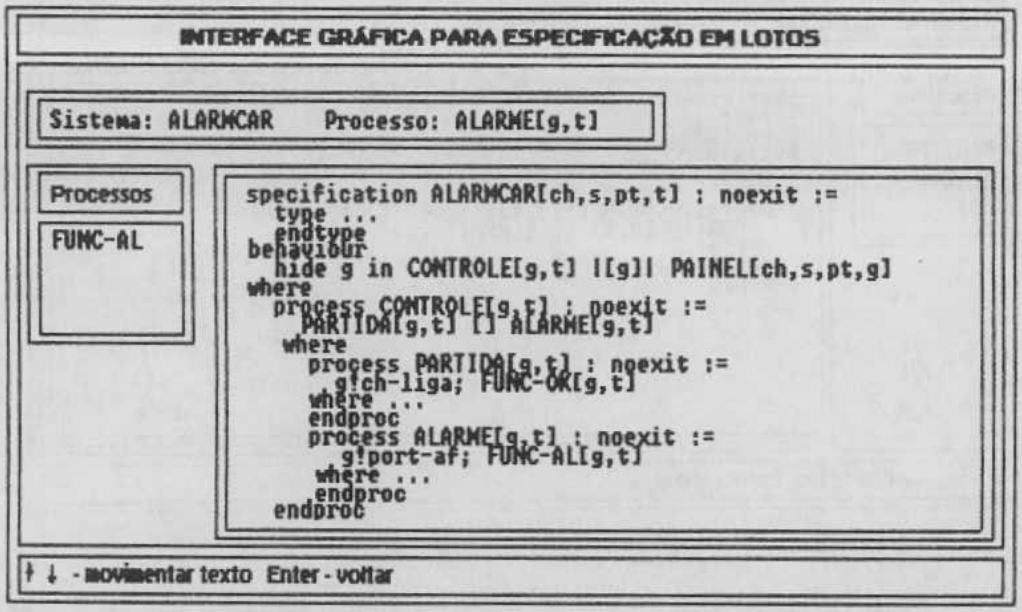
Nesta seção, apresentaremos a utilização da interface no desenvolvimento estruturado do sistema de alarme descrito na seção 3. Para não tornar este exemplo muito extenso, não apresentaremos todas as telas para o desenvolvimento da aplicação; procuraremos, contudo, dar uma idéia geral do funcionamento da interface.

A primeira decomposição do sistema nos processos Controle e Painel foi apresentada na seção anterior. O processo Controle, por sua vez, pode ser decomposto em dois subprocessos, um processo Partida que trata o funcionamento normal do carro e um processo Alarme que trata o funcionamento do alarme. Estes processos não trocam informações e são exclusivos. Através da opção de REPRESENTAÇÃO TEXTUAL COMPLETA podemos visualizar o estado geral do sistema como mostra a figura a seguir.

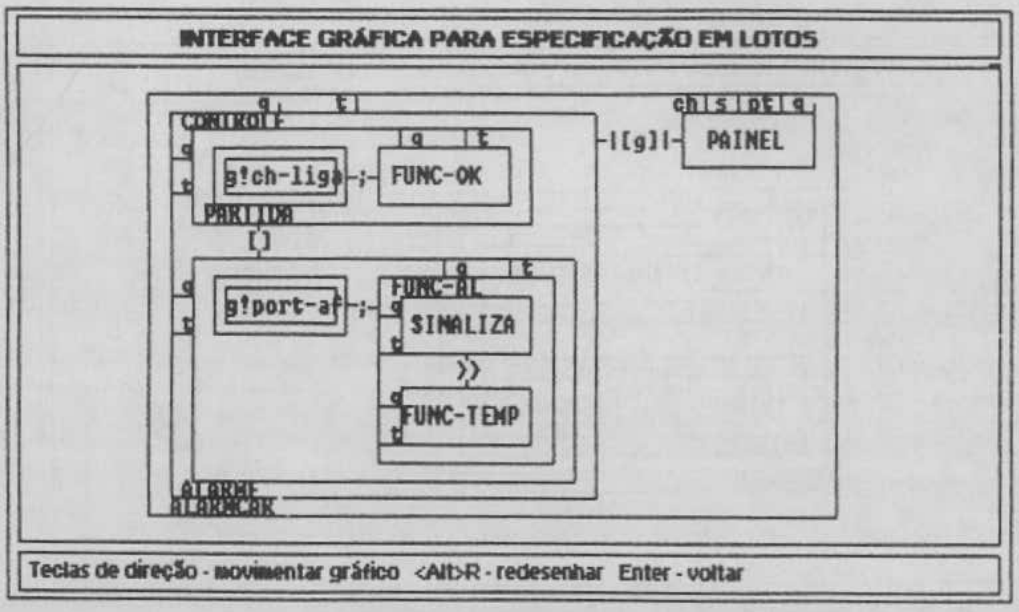


A escolha entre os processos Partida ou Alarme vai depender do evento. Se ocorrer o evento que oferece o sinal liga chave de ignição será escolhido o processo Partida, enquanto que se ocorrer o evento que oferece o sinal abre/fecha porta do carro será escolhido o processo Alarme. Desta forma, para o processo Partida temos um sequenciamento da ocorrência do evento que oferece o sinal liga chave de ignição (g!ch-liga) e o funcionamento do carro (processo Func-OK); e para o processo Alarme temos o sequenciamento da ocorrência do evento que oferece o sinal abre/fecha porta do carro (g!port-af) e o funcionamento do alarme (processo Func-Al). Após os refinamentos nos processos Partida e Alarme as representações gráfica e textual do sistema são visualizadas como segue:

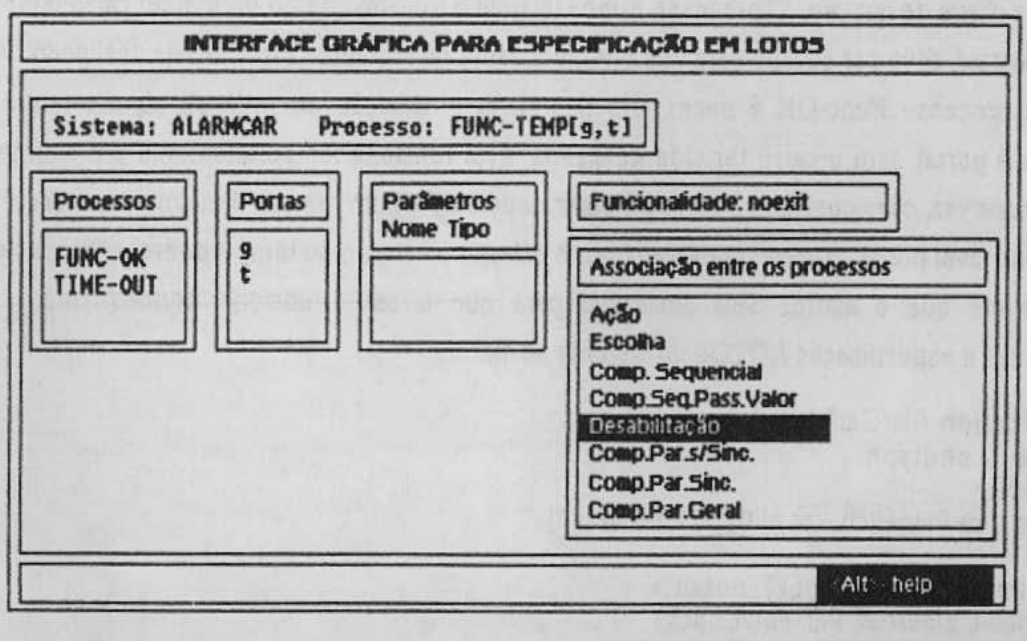




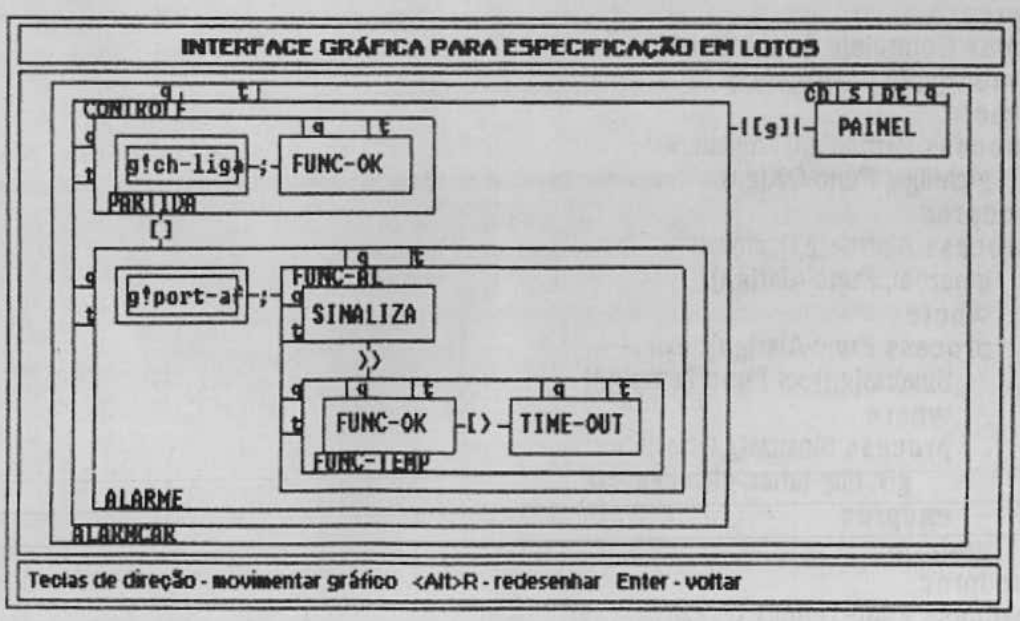
O funcionamento do alarme, conforme descrito anteriormente, requer que o sinalizador seja ativado uma vez, o que faz com que o carro funcione temporiariamente. Assim, podemos decompor o processo Func-Al em dois subprocessos compostos sequencialmente, um processo Sinaliza e um processo Func-Temp. Através da opção de visualização da representa,ão gráfica completa do sistema podemos obter o seguinte gráfico:



Depois do alarme ser sinalizado uma vez, este deverá ser sinalizado novamente, caso contrário, depois de determinado tempo t o carro pára. Assim, o processo Func-Temp pode ser decomposto em dois subprocessos, um processo Func-OK para o funcionamento normal do carro e um processo Time-Out, o qual interrompe o primeiro processo caso ocorra estouro de temporização. A figura abaixo mostra a tela de interação para a introdução deste refinamento no processo Func-Temp, onde é selecionada a operação de desabilitação para representar a associação entre os processos que compõem este refinamento.



A representação gráfica da estrutura geral do sistema pode ser visualizada na figura abaixo.



Finalmente, devemos descrever o comportamento interno dos processos Painel, Sinaliza, Func-OK e Time-Out. O processo Painel simplesmente recebe do ambiente as ativações do sinalizador de alarme, os sinais de liga e desliga a chave de ignição e abre/fecha a porta do carro e os sincroniza com o processo Controle. O processo Sinaliza é responsável pela primeira ativação do sinal de alarme, o que provoca o evento que sincroniza com o ambiente externo a ativação de um contador de temporização, e pela ativação do sinal liga chave de ignição. O processo Func-OK trata o funcionamento normal do carro lembrando que o sinal de alarme deve ser ativado para que o temporizador do ambiente externo seja desligado. Ainda com relação ao processo Func-OK é necessário descrever a situação em que se alguém entra no carro (abre/fecha a porta) sem o carro ter sido desligado, este funciona temporariamente e o alarme deve ser sinalizado uma vez, caso contrário o carro irá parar depois de algum tempo. Finalmente, o processo Time-Out é responsável por sincronizar com o ambiente externo o estouro do temporizador, o que faz com que o carro pare até que o alarme seja sinalizado para que o carro funcione normalmente. A seguir, apresentamos a especificação LOTOS do sistema de alarme:

```
specification AlarCar[ch,s,pt,t] : noexit :=
   type ... endtype
behaviour
   hide g in Painel[ch,s,pt,g] |[g]| Controle[g,t]
where
   process Painel[ch,s,pt,g] : noexit :=
      pt!a-f; g!port-af; Painel[ch,s,pt,g]
   [ ] ch!liga; g!ch-liga; Painel[ch,s,pt,g]
   [ ] ch!desliga; g;ch-desl; Painel[ch,s,pt,g]
   [ ] s; g!s; Painel[ch,s,pt,g]
   endproc
   process Controle[g,t] : noexit :=
      Partida[g,t] [ ] Alarme[g,t]
      where
      process Partida[g,t] : noexit :=
         g!ch-liga; Func-OK[g,t]
      endproc
      process Alarme[g,t] : noexit :=
         g!port-af; Func-Alar[g,t]
         where
         process Func-Alar[g,t] : exit :=
            Sinaliza[g,t] >> Func-Temp[g,t]
            where
            process Sinaliza[g,t] : exit :=
               g!s; t!lig-timer; g!ch-liga; exit
            endproc
         endproc
      endproc
      process Func-Temp[g,t] : exit :=
```

```
        Func-OK[g,t] [> Time-Out[g,t]
        where
        process Time-Out[g,t] : noexit :=
            t!pare; g!s; g!ch-liga; Func-OK
        endproc
    endproc
    process Func-OK[g,t] : noexit :=
        g!s; t!desl-timer; Func-OK[g,t]
    [ ] g!port-af; t!lig-timer; Func-Temp[g,t]
    [ ] g!ch-desl;   ( g!port-af; Controle[g,t] [] Controle[g,t] )
    endproc
endspec
```

5. CONSIDERAÇÕES FINAIS

Neste trabalho, apresentamos as principais características de uma ferramenta de desenvolvimento de especificações, a qual foi desenvolvida para prover uma interface gráfica de fácil adequação ao nosso ambiente de desenvolvimento estruturado de especificações LOTOS. A interface é uma ferramenta simples e de fácil utilização, já que o nosso objetivo principal é uma interação eficiente com o usuário.

A interface permite o desenvolvimento modular da especificação, onde novas características vão sendo adicionadas através de refinamentos sucessivos nos módulos do sistema. Durante o desenvolvimento da especificação é possível visualizar as representações gráfica e textual do sistema, é possível selecionar determinadas partes da especificação e passar de forma rápida do gráfico para o texto.

Procuramos evidenciar também pontos importantes que devem ser considerados num ambiente de desenvolvimento modular, como é a questão do gerenciamento de aspectos relacionados com a consistência das informações. A interface apresentada procura, na medida do possível, checar as informações para detectar e corrigir inconsistência nos dados da especificação como é a questão de manter a estrutura do refinamento.

Outro aspecto importante é o gerenciamento da representação gráfica na interface. Apresentamos aqui algumas idéias do algorítmo de tratamento do diagrama com o intuito de permitir ao usuário uma forma se não ideal pelo menos eficiente de visualizar graficamente a estrutura geral do sistema. Em particular, uma ferramenta de *zoom*, a qual pretendemos implementar com a finalidade de expandir a capacidade atual deste nosso ambiente, serial ideal para tratar esta questão.

Com a implementação deste ambiente, a especificação gerada a partir da interface poderia ser integrada à ferramentas de verificação e execução ou simulação de especificações LOTOS, compondo um ambiente completo para execução desta linguagem. Em particular, um trabalho posterior a este ambiente aqui proposto seria a integração da nossa ferramenta a um ambiente de simulação de especificações LOTOS utilizando linguagens funcionais como proposto em [FeCun 89], [Ferraz 89].

REFERÊNCIAS BIBLIOGRÁFICAS

[BoBri 87] Bolognesi,T. e Brinksma,E.: "Introduction to the ISO Specification Language LOTOS", **Computer Networks and ISDN Systems**, vol. 14, 1987, pp. 25-59.

[Brink 86] Brinksma,E.: "A Tutorial on LOTOS", **Protocol Specification, Testing and Verification, V**, editado por M. Diaz, North Holland, pp. 171-194, 1986.

[Cunha 81] Cunha,P.R.F.: "Design and Analysis of Message Oriented Programs", **Tese de Doutorado, Universidade de Waterloo**, Canadá, 1981.

[DeSch 86] Delise,N e Schwartz,M.: "A Programming Environment for CSP", **ACM**, 1986, pp. 34-41.

[EhMha 85] Ehrig,H. e Mhar,B.: "Fundamentals of Algebraic Specification 1", **Springer-Verlag**, Berlim, 1985.

[Ferraz 89] Ferraz,C.A.G.: "Um Estudo para o Desenvolvimento de Protótipos de Especificações LOTOS usando Programação Funcional", **Dissertação de Mestrado, Departamento de Informática**, Universidade Federal de Pernambuco, Dez. 1989.

[FeCun 89] Ferraz,C.A.G e Cunha,P.R.F.: "Simulação de Especificações LOTOS usando LinguagensFuncionais", **III Simpósio Brasileiro de Engenharia de Software**, Recife, Out. 1989.

[ISO 88] ISO IS 8807: "LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour", Maio 1988.

[KrMNg 89] Kramer,J., Magee,J. e Ng,K.: "Graphical Support for Configuration Programming", **Departamento de Computação, Imperial College**, Jan. 1989.

[Milne 80] Milner,R.: "A Calculus of Communication Systems", **Lecture Notes in Computer Science**, Spring-Verlag, 1980.

[Moura 89] Moura,M.T.S.: "Desenvolvimento Estruturado de Especificações em LOTOS", **Dissertação de Mestrado, UFPE**, Dez. 1989.

[MoCun 89] Moura,M.T.S. e Cunha,P.R.F: "Representação Gráfica para LOTOS", **III Simpósio Brasileiro de Engenharia de Software**, Recife, Out. 1989.

[Myers 78] Myers,G.J.: "Composite/Structured Design", **van Nostrand Reinhold Co.**, 1978.

[YuNak 85] Yuasa,T. e Nakagima,R.: "IOTA: A Modular Programming System", **IEEE Transaction on Software Engineering**, vol. SE-11, no. 2, Fev. 1985.