COPPEAD/UFRJ

RELATÓRIO COPPEAD Nº 228

GERSIMUL: UM GERADOR AUTOMÁTICO DE PROGRAMAS DE SIMULAÇÃO EM TURBO PASCAL

Eduardo Saliby (*)
Gilson Carvalho de Oliveira (**)

Agosto 1989

(*) Professor e pesquisador da COPPEAD - Instituto de Pós-Graduação e Pesquisa em Administração / UFRJ.
PhD em Pesquisa Operacional (Lancaster, 1980),
Mestre em Engenharia de Produção (COPPE/UFRJ, 1974).

(**) Mestrando em Engenharia de Sistema e Computação - Pesquisa Operacional pelo Instituto Militar de Engenharia (IME).
Engenheiro Eletricista (IME), Ten Cel QEM.

RESUMO

Este trabalho apresenta o conceito de geração automática de programas de simulação, que tem por objetivo minimizar a necessidade de programação em estudos de simulação. A seguir, com base em uma possível estrutura de um programa gerador, descreve-se o GERSIMUL. O GERSIMUL, gerador automático de programas de simulação para o sistema SIMUL, dispõe de facilidades para a modelagem interativa, edição de modelos e geração de programas em linguagem TURBO PASCAL.

Uma descrição do uso do GERSIMUL, estruturas de dados utilizadas, arquivos de entrada e arquivos gerados são apresentados.

1 . INTRODUÇÃO

O objetivo deste trabalho é descrever de forma sucinta os principais passos envolvidos no desenvolvimento de um programa para a geração automática de programas de simulação.

O desenvolvimento do sistema SIMUL (Pimentel, 1989) para a simulação a eventos discretos resultou numa biblioteca de rotinas pré-definidas para serem utilizadas em conjunto com uma estrutura de programa padrão, definida pelo seu módulo executivo. Sendo programado em TURBO-PASCAL, o usuário se encarregaria de preencher o módulo executivo de acordo com as peculiaridades do problema em estudo.

Apesar do SIMUL simplificar bastante o trabalho de programação de uma simulação, permanece ainda a necessidade de o usuário programar parte do modelo. Esta exigência, certamente, inibe uma utilização mais ampla do sistema.

Uma forma de se contornar este problema é através de um gerador automático de programas que substitui o usuário nesta tarefa de preenchimento do módulo executivo. Neste caso, caberia ao usuário fornecer apenas as informações que definem o modelo a ser simulado, num ambiente interativo.

2 . CONCEITOS BÁSICOS

2.1-SIMULAÇÃO

Simulação é aqui entendida como a técnica de Pesquisa Operacional que permite o estudo de problemas do mundo real que dificilmente teriam, ou mesmo não teriam, possibilidade de solução analítica.

O presente trabalho refere-se à simulação a eventos discretos de natureza probabilística. Os eventos discretos correspondem às mudanças de estado do sistema que ocorrem em instantes específicos ao longo do tempo. Por exemplo, na simulação de um bar, as chegadas e saídas de clientes determinam mudanças de estado do sistema que ocorrem em pontos bem determinados ao longo do tempo.

Um estudo de simulação envolve duas atividades básicas: a modelagem do problema a ser simulado e a simulação computacional propriamente dita.

2.2-A MODELAGEM DO PROBLEMA A SER SIMULADO

A modelagem do problema a ser simulado é a atividade de formulação e análise de um modelo que represente um problema do mundo real.

O encarregado da modelagem pode se apoiar nas seguintes fontes (Chew, 1986) para definir o modelo:

i -discussão com "especialistas" no sistema a ser simulado, como gerentes ou trabalhadores, que entendem de diferentes aspectos do sistema.

ii -observação do sistema e/ou estudo de relatórios escritos sobre a operação de um sistema existente ou proposto.

iii -análise estatística de dados numéricos do sistema e/ou

sistemas similares.

iv -revisão de modelos relacionados já desenvolvidos, incluindo teoria relevante.

v -intensa reflexão.

O processo de modelagem pode ser resumido pelo fluxograma (Chew, 1986) da Figura 1.

O modelo de simulação tem que ser formulado em algum tipo de linguagem simbólica que capture a lógica do problema a ser simulado. Uma dessas linguagens, que tem sido bastante utilizada, é o Diagrama de Ciclos de Atividades (DCA) (Pidd, 1988). O DCA é descrito no item 2.4.

O modelo de simulação, para ser transformado em um programa de computador, tem que ser estruturado de alguma forma. Uma das abordagens de modelagem que conduz a programas bem estruturados é o "método das três fases" (Crookes, 1986), que leva a uma estrutura de programa bastante "transparente".

2.3-SIMULAÇÃO COMPUTACIONAL

Simulação computacional é a atividade de codificação e processamento de um progrma, descrevendo o problema a ser simulado, em computador. É, portanto, identificada como a parte experimental do estudo.

Se for considerado que a atividade de modelagem da simulação termina com a descrição lógica do problema através de uma linguagem simbólica adequada (no caso o DCA), a próxima atividade necessária é a codificação, ou seja, a transformação do modelo lógico em um programa codificado em uma determinada linguagem, que possa ser processada por computador. A codificação poderá ser

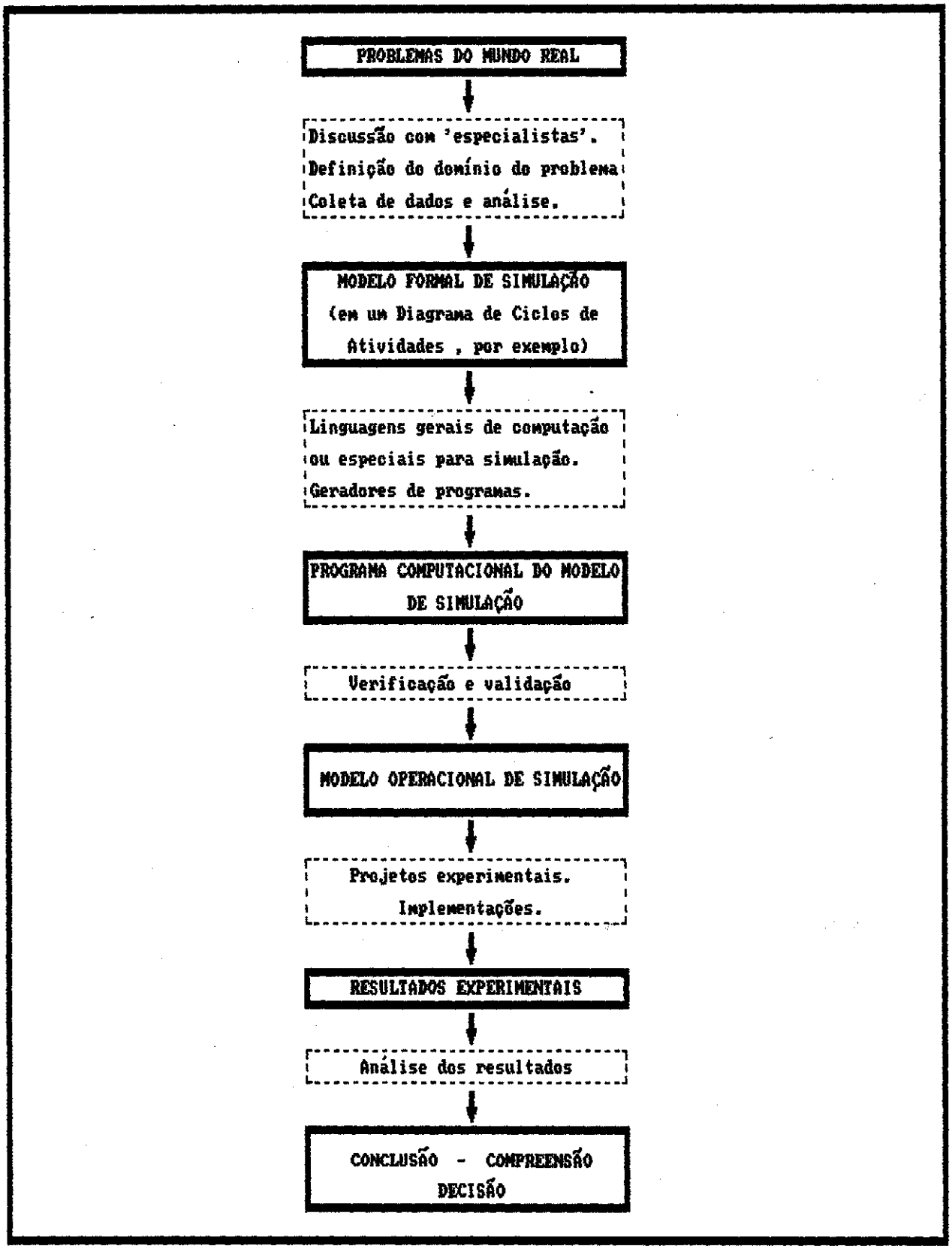


Figura 1 - O Processo de Modelagem de Simulação.

feita em uma linguagem de uso geral (FORTRAN,PASCAL etc) ou em uma linguagem especial para simulação (GPSS, SIMSCRIPT etc).

Utilizando-se uma linguagem de uso geral, a tarefa de codificação de um programa de simulação será simplificada caso se disponha de um conjunto de rotinas pré-definidas. Sistemas de simulação que oferecem este tipo de facilidade são o eLSE, o VSS e o SIMUL.

2.4-O DIAGRAMA DE CICLOS DE ATIVIDADES

O Diagrama de Ciclos de Atividades (DCA) pode ser visto como uma linguagem simbólica que permite capturar a lógica de um problema a ser simulado (Pidd, 1988). Basicamente, apenas dois símbolos são utilizados para mostrar como as entidades envolvidas em um problema ou sistema interagem entre si :

Para um DCA, cada classe de entidade tem um ciclo de vida individual. Este ciclo consiste de uma série de estados alternantes entre estados ativos (quando a entidade está envolvida em uma atividade) e estados passivos (quando a entidade está em uma fila aguardando condições para participar de uma atividade).

No caso de não ser caracterizado um estado passivo entre duas atividades, ainda assim é criada uma fila, falsa ("dummy"), com o objetivo de atender os requisitos lógicos do DCA.

O exemplo do DCA do Anexo 3, feito para o problema do bar descrito no Anexo 2, mostra como um DCA é elaborado. Neste DCA aparece um terceiro símbolo que representa uma "fonte" e

"sumidouro" de entidades.

3 . GERADORES AUTOMATICOS DE PROGRAMAS DE SIMULAÇÃO

Considerando que um determinado problema do mundo real tenha sido corretamente modelado, com sua lógica descrita em uma linguagem adequada (DCA, por exemplo), e tendo sido escolhida a lin-linguagem de programação a ser utilizada, o próximo passo, como foi visto, é a codificação do programa de simulação. Mesmo com o uso de rotinas pré-definidas, o processo de codificação é trabalhoso e pouco criativo, particularmente para os problemas mais complexos. Além disso, durante a codificação podem ser introduzidos erros no programa.

Os geradores automáticos de programas de simulação propiciam um modo de se superar essas inconveniências e ainda outras vantagens.

Formalmente, os geradores automáticos de programas podem ser descritos como uma interface que aceita a descrição lógica do problema a ser simulado na forma de um diagrama simbólico simples (DCA, por exemplo) e a converte em um programa de simulação numa determinada linguagem alvo (Kienbaum, 1988).

3.1-VANTAGENS E DESVANTAGENS DOS GERADORES AUTOMATICOS

Como principais vantagens no uso de geradores automáticos de programas de simulação pode-se citar:

-o tempo de elaboração do programa de simulação reduz-se significativamente, visto que é praticamente o tempo gasto na entrada interativa de dados, uma vez que é desprezível o tempo gasto pelo computador na geração do programa. Este fato encoraja a criação de protótipos e facilita a revisão ou manutenção do modelo.

-a interface de modelagem pode incluir facilidades que propiciem um rigoroso teste de lógica do modelo e testes de sintaxe nas fórmulas fornecidas pelo usuário, eliminando possíveis erros na entrada de dados e produzindo um programa inteiramente coerente com a descrição do modelo fornecida. Ao contrário de um programa escrito manualmente, o programa gerado é praticamente isento de erros.

-o programa gerado é "transparente" devido à estruturação uniforme (face a ser gerado automaticamente) e à boa documentação normalmente apresentada.

As principais desvantagens geralmente citadas para programas gerados automaticamente são:

-há uma tendência dos programas serem de processamento mais lento, devido à alta estruturação apresentada pelos mesmos.

-limitações relativas à representação de modelos grandes e complexos.

Quanto ao fato dos programas gerados automaticamente serem intrinsicamente de processamento mais lento, este é o preço que se paga pela "transparência" do programa. Com a tendência dos computadores se tornarem cada vez mais rápidos, esta restrição tem cada vez menor peso.

As limitações relativas à representação de modelos grandes e complexos está mais ligada a limitações da linguagem simbólica utilizada na descrição do modelo do que propriamente à geração automática em si. Esta desvantagem é compensada pela facilidade com que o programa gerado pode ser modificado ou complementado para se obter uma melhor aderência ao problema real.

Cabe mencionar, no entanto, que estão sendo desenvolvidos es-

forços no sentido de aperfeiçoar as linguagens simbólicas de descrição do modelo, em particular o DCA.

3.2-ESTRUTURA DE UM GERADOR DE PROGRAMAS

Um gerador de programas pode ser estruturado conforme o mostrado na Figura 2 (Kienbaum, 1988-adaptado). Nesta figura os blocos representam:

A -módulo de edição e análise do modelo.

Este módulo é uma interface que permite a entrada dos dados que representam o modelo a ser simulado e faz uma crítica ou verificação lógica dos dados fornecidos, verificando inclusive a sintaxe de fórmulas e funções da linguagem em que o programa será gerado. Após a conclusão da entrada de dados é criado um arquivo (1), com dados estruturados, que contém todas as informações necessárias à geração do programa. Este arquivo pode ser considerado como parte da documentação do programa a ser criado.

B -módulo de geração.

Este módulo faz a geração do programa de simulação. Ele usa um arquivo (2) que contém a estrutura padrão do programa a ser gerado (que independe do programa em particular) e do arquivo (1) com os dados do programa especifico a ser gerado. O programa gerado vai para o arquivo (3). O arquivo (4) recebe valores de parâmetros utilizados unicamente na fase de execução (extraídos de (1)),como por exemplo, o tamanho das filas de espera no inicio da simulação. A vantagem de se separar estes parâmetros em um arquivo especial é que, desta forma, torna-se possivel a

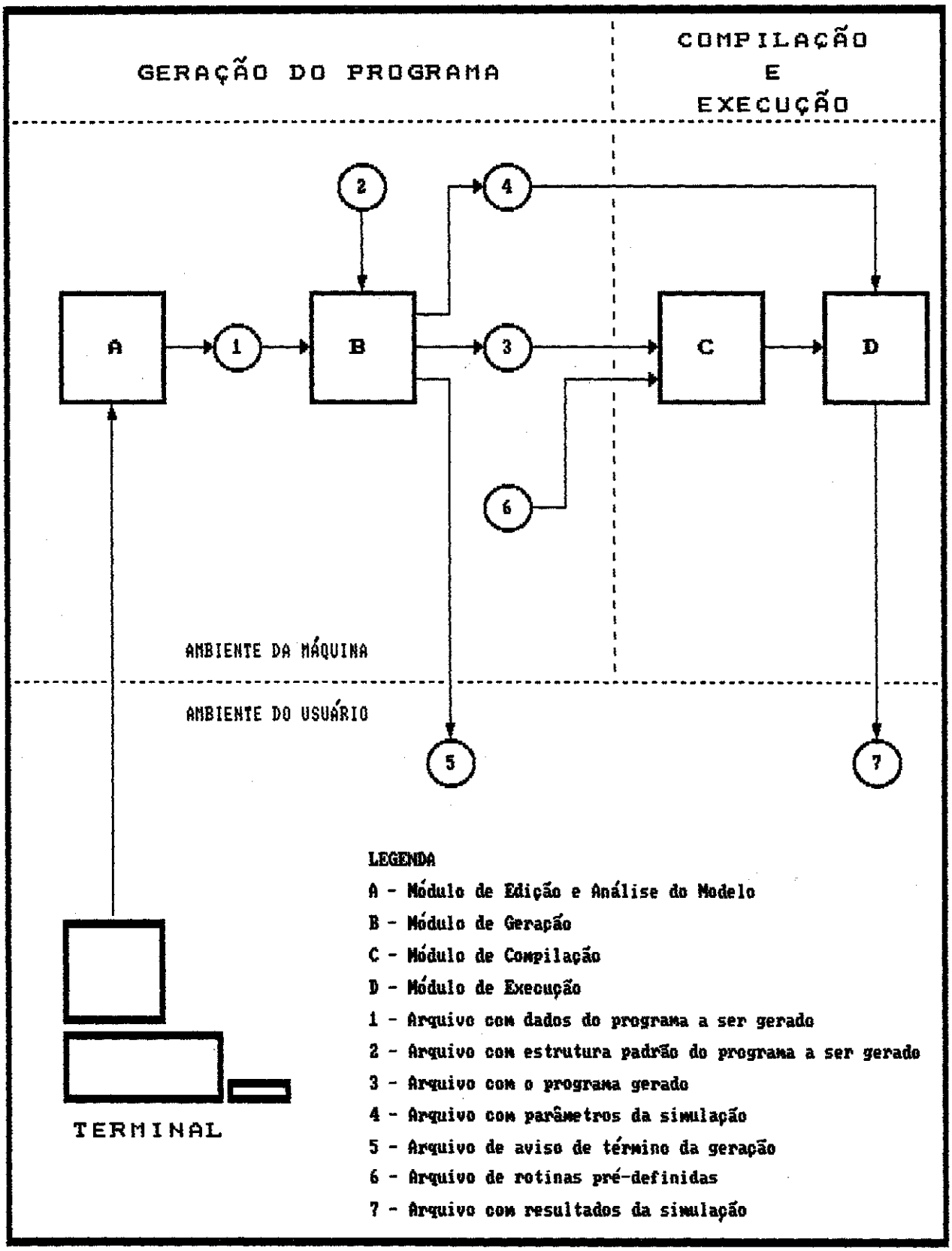


Figura 2 - Estrutura de um Gerador de Programas de Simulação.

modificação destes parâmetros sem que se altere o programa gerado. Esta facilidade é particularmente útil para a comparação de alternativas de operação do sistema e para análises de sensibilidade. O arquivo (5) serve para avisar o usuário do término da geração e é usualmente a própria tela do terminal.

C -módulo de compilação.

É o módulo onde o programa gerado (guardado no arquivo (3)) é compilado. Este módulo poderá utilizar uma biblioteca de rotinas pré-compiladas guardadas no arquivo (6).

D -módulo de execução.

Módulo onde o programa gerado é executado,sendo os resultados guardados no arquivo (7).

4 . O GERSIMUL

4.1-INTRODUÇÃO

GERSIMUL é o nome de um gerador automático de programas de simulação que está sendo desenvolvido utilizando como base a biblioteca de rotinas do SIMUL (Pimentel, 1989).

O GERSIMUL foi inspirado no pacote de simulação VS6 (Syspack, 1988) desenvolvido originalmente pelo grupo CASM da London School of Economics. O GERSIMUL está estruturado conforme o diagrama da Figura 2 e foi escrito na linguagem Turbo Pascal 5.0, para a utilização em micros PC compatíveis.

4.2-DESCRIÇÃO GERAL DO GERSIMUL

O GERSIMUL tem uma interface de modelagem (correspondente ao bloco A da Figura 2) através da qual o usuário, em uma sessão interativa, entra com os dados do problema a ser simulado, a partir do DCA que o descreve.

Após a sessão de entrada de dados os mesmos são "salvos" em um arquivo (1) - arquivo "nome.SIM".

Para a geração do programa (bloco B da Figura 2), os dados do problema guardados em (1) são recuperados através da leitura desse arquivo e "carregamento" dos seus dados em listas encadeadas formadas na memória RAM do computador. Após os dados estarem carregados é feita a geração propriamente dita do programa de simulação, com a utilização de um arquivo (2) que contém a estrutura padrão do programa a ser gerado - arquivo "SUPORTE.PAS" (Anexo 5). Ao gerar o programa,o módulo gerador também gera um arquivo (4) que contém as informações dos estados iniciais de todas as filas, com exceção das filas vazias, do problema a ser simulado -

arquivo "nome.QUE".

A não ser em casos muito particulares, a prática recomenda que se inicie uma simulação pelo método das três fases com todas as entidades em filas, ou seja, com nenhuma atividade em andamento. Assim sendo, a informação do arquivo (4) define o número de entidades presentes no sistema quando do início da simulação. Como conseqüência, esta é a forma pela qual o número de entidades permanentes de cada tipo é definida.

O "arquivo" (5) corresponde a uma mensagem na tela do computador que anuncia quando a geração do programa foi completada.

Após o programa ter sido gerado e colocado no arquivo (3) - arquivo "nome.PAS", ele pode ser compilado com o auxílio das rotinas pré-definidas do SIMUL e então executado, sendo os resultados da simulação guardados em (7) - arquivo "nome.REP".

Na realidade, o GERSIMUL pode operar de um modo um pouco mais "esperto" do que o acima descrito: após a modelagem, não é necessário criar o arquivo (1) com os dados de entrada do programa a ser gerado, para sua posterior recuperação. Numa mesma sessão,ao término da modelagem, os dados do problema já estão disponíveis na memória do computador, podendo-se seguir imediatamente para a geração do programa. A utilidade do arquivo (1) reside na possibilidade de recuperação dos dados de problemas modelados em sessões anteriores e, principalmente, na facilidade de edição destes dados, permitindo assim modificações do modelo utilizando os recursos interativos da interface de modelagem (A).

4.3-A MODELAGEM

A modelagem do problema a ser simulado é feita através de

"janelas" apresentadas na tela do computador. O usuário, a partir do DCA que descreve o problema, digita os dados solicitados nas "janelas".

Para guardar os dados de entrada, é utilizada uma estrutura de listas encadeadas e fechadas, implementadas através de "registros e ponteiros" em Pascal, conforme pode ser visto no Anexo 1.

Para cada janela aberta, o programa cria uma estrutura correspondente que recebe os dados digitados pelo usuário. Ao término da sessão de modelagem, todos os dados necessários à geração do programa estão disponíveis na memória do computador, podendo então ser salvos no arquivo (1) - "nome.SIM".

4.4-O CARREGAMENTO

Os dados do problema a ser simulado são lidos no arquivo (1)- "nome.SIM"- e carregados na memória RAM do computador através da "unit" CARREGAR do GERSIMUL. Para isso, é utilizada a mesma estrutura de listas encadeadas e fechadas da interface de modelagem.

A explicação de como a "unit" CARREGAR funciona seria muito extensa, contudo o acompanhamento do problema do BAR, no Anexo 2, dá uma idéia geral de como o programa trabalha.

O Anexo 2 traz a descrição do problema do BAR, sendo que o DCA correspondente é apresentado no Anexo 3. No Anexo 4 está listado o arquivo BAR.SIM, criado pela interface de modelagem, que apresenta todos os dados do DCA de forma estruturada.

O arquivo BAR.SIM é organizado em três partes, delimitadas por uma linha de "=". A primeira traz informações sobre as "entidades" do sistema: nome das entidades, suas quantidades,se

são originadas por "fontes" ou não, e se têm ou não "atributos". Por exemplo, a entidade CLIENTE tem quantidade "infinita" (∞) já que provém da fonte RUA e um atributo chamado SEDE.

As informações referentes às entidades são recuperadas e "colocadas" na memória do computador com a utilização dos ponteiros e registros "entidade","fonte" e "atributo" apresentadas no Anexo 1.

A segunda e terceira parte do arquivo BAR.SIM contém, respectivamente, informações relativas às "filas" e "atividades" envolvidas no problema. Essas informações são recuperadas de forma similar à já explicada.

Quando a fase de "carregamento" é concluída, todas as informações relativas ao programa a ser gerado estarão disponíveis em listas encadeadas fechadas na memória do computador.

Observe-se que algumas estruturas apresentadas no Anexo 1 não são utilizadas no problema do BAR, já que são destinadas a problemas mais complexos.

4.5-A GERAÇÃO

A geração do programa é feita pela "unit" GERAR do GERSIMUL. Esta unidade basicamente lê o arquivo SUPORTE.PAS que traz a estrutura padrão do programa a ser gerado (Anexo 5) e, em pontos identificados pelo símbolo #, entra com os "comandos", "procedures" etc pertinentes. As informações necessárias à construção desses comandos,procedures etc são retiradas das listas encadeadas na memória do computador.

O Anexo 6 mostra a listagem do programa gerado para o BAR.

A "unit" GERAR também cria um outro arquivo (BAR.QUE) que

guarda as informações sobre o estado das filas no início da simulação (Anexo 7).

4.6-A COMPILAÇÃO E A EXECUÇÃO

A compilação e a execução do programa gerado pode então ser feita com o auxílio das rotinas pré-definidas do SIMUL. É produzido um relatório com os resultados da simulação, que é guardado em um arquivo com o nome BAR.REP.

4.7-RESUMO DAS FACILIDADES APRESENTADAS PELO GERSIMUL

Conforme o exposto nos itens anteriores, o GERSIMUL apresenta as facilidades de:

- MODELAR um problema de simulação através de uma interface interativa, onde a entrada de dados é feita a partir do DCA que descreve o problema.

- SALVAR a descrição lógica do problema a ser simulado em um arquivo estruturado que contenha todos os dados fornecidos através da interface de modelagem.

- CARREGAR um modelo previamente modelado e salvo, para possíveis modificações ou geração do programa.

- EDITAR, que corresponde à possibilidade de se modificar o modelo com o qual se está trabalhando, ou seja, o modelo carregado na memória.

- GERAR um programa de simulação que possa ser compilado com o auxílio de rotinas do SIMUL e posteriormente executado.

5 . CONSIDERAÇÕES FINAIS

O GERSIMUL é um programa modular que se encontra em fase de testes. Em uma análise preliminar, ele confirma todas as vantagens citadas no item 3.1 relativas aos geradores automáticos.

Futuramente, prevê-se a adição de novas facilidades ao GERSIMUL, tais como a execução da simulação sem geração prévia do programa, visualização da simulação e a análise mais completa dos seus resultados.

O desenvolvimento do GERSIMUL tem sido uma experiência gratificante não só com respeito à criação de um gerador automático, mas também quanto às possibilidades da utilização da linguagem Pascal com esse propósito.

O GERSIMUL é uma complementação do sistema SIMUL, tornando sua utilização mais simples e rápida. O GERSIMUL, junto com os demais módulos ora em desenvolvimento, transformará o SIMUL em um autêntico "ambiente integrado de simulação".

REFERÊNCIAS

CHEW, S. T. Program generator for discrete event digital simulation modelling. London, London School of Economics, 1986. Tese de Doutorado.

CROOKES, J. G. et allii. A three-phase simulation system written in Pascal. Journal of Operational Research Society, 37 (6): 603-18, June 1986.

KIENBAUM, G. S. Uma nota sobre a incorporação de geradores de progamas a ambientes integrados de auxílio à modelagem e simulação discretas. São José dos Campos, Laboratório de Computação e Matemática Aplicada / INPE, 1988. (Relatório Técnico).

PIDD, M. Computer simulation in management science. 2. ed. Chichester, J. Wiley, 1988.

PIMENTEL, M. Sistema computacional para simulação discreta com opção de uso da amostragem descritiva. Rio de Janeiro, Instituto Militar de Engenharia, 1989.

SYSPACK LTD. The VS6 user's guide. London, Syspack, 1988.

- ANEXO 1 -

ESTRUTURA DE DADOS USADAS PELO GERSIMUL

```
type
  str4  = string[4];
  str10 = string[10];
  str20 = string[20];
  str80 = string[80];
  hist_tipo = (Tamanho,Tempo);


  nome = ^nomelista;
  nomelista = record
                x1,y1 : integer;
                nome  : str20;
                px    : nome;
              end;


  atrib_histograma = ^atrhlista;
  atrhlista = record
                nome    : str10;
                base    : integer;
                largura : integer;
                px      : atrib_histograma;
              end;


  atributo = ^atriblista;
  atriblista = record
                 nome   : str10;
                 hist   : atrib_histograma;
                 y      : integer;
                 pxd    : atributo;
                 pxe    : atributo;
               end;


  entidade = ^entlista;
  entlista = record
               nome   : str10;
               num    : integer;
               fonte  : str10;
               atrib  : atributo;
               y      : integer;
               pxe    : entidade;
               pxd    : entidade;
             end;
```

```
fila_histograma = ^fila_histlista;
fila_histlista = record
                   nome     : str10;
                   base     : integer;
                   largura  : integer;
                   h_tipo   : hist_tipo;
                   y        : integer;
                   pxe      : fila_histograma;
                   pxd      : fila_histograma;
                 end;

fila = ^filalista;
filalista = record
              nome        : str10;
              ent_fil     : entidade;
              hist        : fila_histograma;
              disciplina  : str4;
              num_inic    : integer;
              num_final   : integer;
              y           : integer;
              pxe         : fila;
              pxd         : fila;
            end;

fonte = ^fontelista;
fontelista = record
               nome : str10;
               ent  : entidade;
               px   : fonte;
             end;


cond_fila_pos = ^condlista;
condlista = record
              condicao    : str80;
              disciplina  : str4;
              fila_pos    : fila;
              y           : integer;
              pxe         : cond_fila_pos;
              pxd         : cond_fila_pos;
            end;


lista_ent_env = ^lista_entlista;
lista_entlista = record
                   apt_e     : entidade;
                   bn        : integer;
                   min       : integer;
                   max       : integer;
                   fila_ant  : fila;
                   fila_pos  : cond_fila_pos;
                   y         : integer;
                   pxe       : lista_ent_env;
                   pxd       : lista_ent_env;
                 end;
```

```
atrs_dados = ^atrs_dadoslista;
atrs_dadoslista = record
                    atrib : atributo;
                    prior : integer;
                    nent  : integer;
                    valor : str80;
                    px    : atrs_dados;
                  end;


cond_atr = ^cond_atrlista;
cond_atrlista = record
                  condicao : str80;
                  prior    : integer;
                  atrs     : atrs_dados;
                  px       : cond_atr;
                end;


atividade = ^ativlista;
ativlista = record
              nome    : str10;
              duracao : str80;
              ent_env : lista_ent_env;
              atr_atv : cond_atr;
              prior   : integer;
              y       : integer;
              pxe     : atividade;
              pxd     : atividade;
            end;


series_temp = ^stlista;
stlista = record
            nome      : str10;
            intervalo : integer;
            base      : integer;
            largura   : integer;
            fila_env  : fila;
            px        : series_temp;
          end;
```

- ANEXO 2 -

DESCRIÇÃO DO PROBLEMA DO BAR

Neste bar imaginário, ao chegar um CLIENTE da RUA (atividade CHEGADA), ele fica aguardando ser servido na fila de ESPERA. Os clientes chegam conforme uma distribuição exponencial com média 10 minutos e recebem ao chegar um atributo chamado SEDE. Este atributo determina, a priori, o número de copos de cerveja que cada cliente irá beber e é dado por uma distribuição inteira-uniforme com limites 1 e 4.

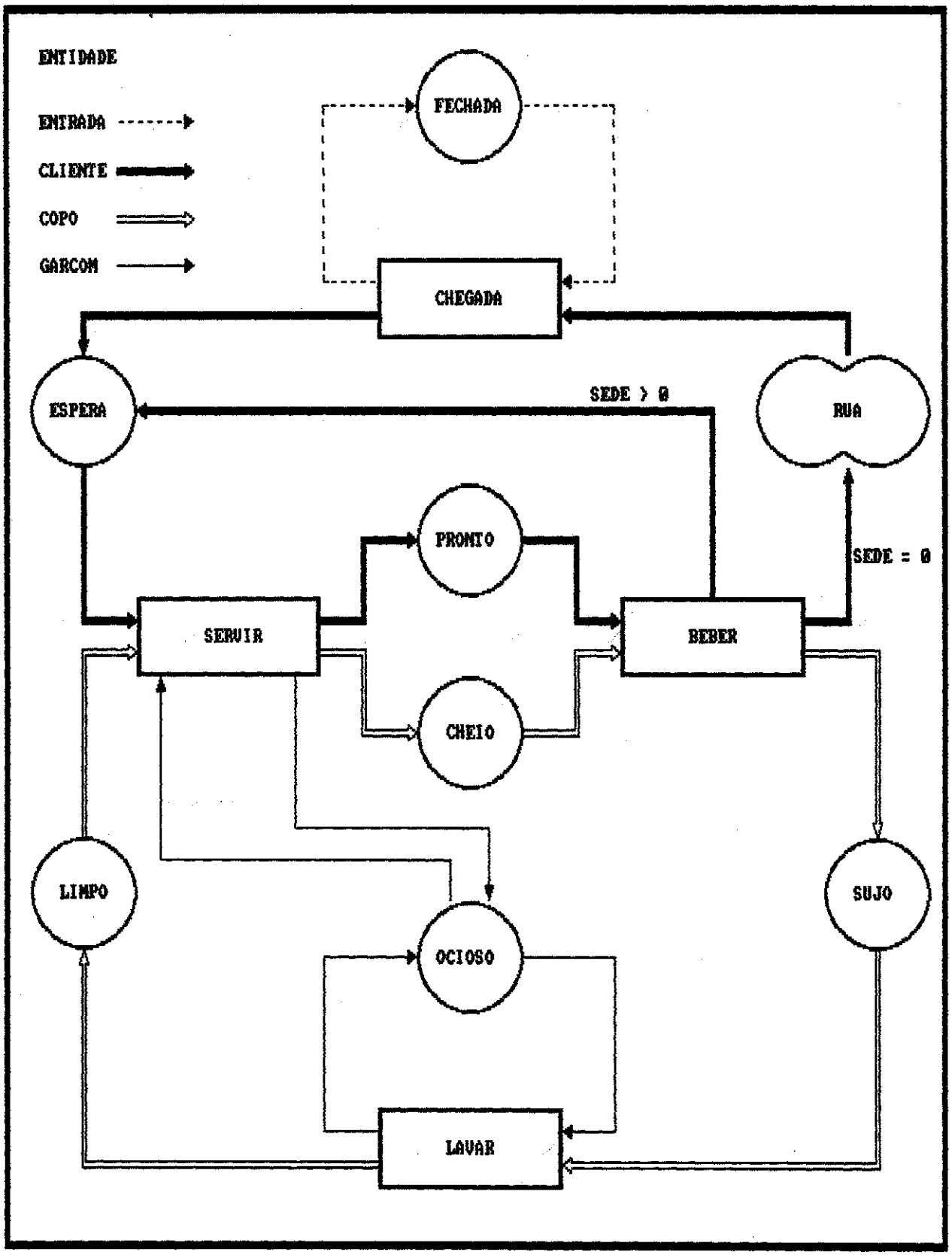
Havendo um COPO na fila LIMPO e um GARÇOM disponível na fila OCIOSO, o CLIENTE é atendido em SERVIR, atividade que tem a duração dada por uma distribuição normal com média 6 minutos e desvio padrão de 1 minuto. Após o término da atividade SERVIR, o CLIENTE vai para a fila PRONTO (fila "dummy") e em seguida passa a BEBER sua cerveja. Concomitantemente, o COPO vai para a fila CHEIO (fila "dummy") e depois vai para a atividade BEBER. O tempo necessário para a atividade BEBER é dado por uma distribuição inteira-uniforme com limites 5 e 8 minutos. O GARÇOM, após SERVIR volta para a fila OCIOSO. Se após BEBER, o CLIENTE tiver sua sede saciada, ele vai embora para a RUA. Caso contrário, ele volta à fila de ESPERA. O COPO, uma vez usado, vai para a fila SUJO, onde aguarda que um GARÇOM fique disponível em OCIOSO para ir para a atividade LAVAR, atividade esta que leva 5 minutos. Uma vez lavado, o COPO vai para a fila LIMPO e o GARÇOM volta à fila OCIOSO.

ENTRADA é uma entidade lógica destinada a "disciplinar" a chegada de clientes no bar, permitindo a entrada de apenas um por vez.

O Anexo 3 mostra o DCA correspondente a este problema.

ANEXO 3

DIAGRAMA DE CICLOS DE ATIVIDADES DO BAR



- ANEXO 4 -

ARQUIVO BAR.SIM

```
PORTA 1 NENHUMA
NIL
SISTEMA 1 NENHUMA
NIL
COPO 70 NENHUMA
NIL
CLIENTE ∞ RUA
SEDE 1
0
NIL
GARCOM 3 NENHUMA
NIL
===========
SUJO FIFO 0 0
COPO
NIL
RUA FIFO ∞ ∞
CLIENTE
NIL
FECHADA FIFO 1 1
PORTA
NIL
ESPERA FIFO 0 0
CLIENTE
T_ESPERA 1 N/A 10 5
F_ESPERA 0 N/A 0 1
NIL
LIMPO FIFO 1 70
COPO
F_LIMPO 0 N/A 0 5
NIL
CHEIO FIFO 0 0
COPO
NIL
PRONTO FIFO 0 0
CLIENTE
NIL
OCIOSO FIFO 1 3
GARCOM
NIL
===========
CHEGADA
NEGEXP(10,2)
PORTA 1 1
FECHADA
DEFAULT
FIFO FECHADA
NIL
CLIENTE 1 1
RUA
DEFAULT
```

```
FIFO ESPERA
NIL
NIL
NENHUMA
1
SEDE(*) 1
SEDE -1
INTEIRA_UNIFORME(1,4,11)
ATRIB_SEDE 8 N/A 0 1
NIL
NIL
NIL
SERVIR
NORMAL(6,1,5)
CLIENTE 1 1
ESPERA
DEFAULT
FIFO PRONTO
NIL
GARCOM 1 1
OCIOSO
DEFAULT
FIFO OCIOSO
NIL
COPO 1 1
LIMPO
DEFAULT
FIFO CHEIO
NIL
NIL
NIL
BEBER
INTEIRA_UNIFORME(5,8,7)
COPO 1 1
CHEIO
DEFAULT
FIFO SUJO
NIL
CLIENTE 1 1
PRONTO
SEDE>0
LIFO ESPERA
DEFAULT
FIFO RUA
NIL
NIL
NENHUMA
1
SEDE(1) 1
SEDE 1
SEDE(1)-1
NIL
NIL
NIL
LAVAR
```

```
5
COPO 1 1
SUJO
DEFAULT
FIFO LIMPO
NIL
GARCOM 1 1
OCIOSO
DEFAULT
FIFO OCIOSO
NIL
NIL
NIL
==========
==========
```

- ANEXO 5 -

ARQUIVO SUPORTE.PAS

```
(** Programa de Simulacao em Pascal 5.0 gerado pelo GERSIMUL. **)

# program nome;
Uses
  Crt,
  Simul;

var
#

procedure Define_Modelo;
begin
#
end;

procedure Atraves_C_Eventos;forward;

procedure Cria_Registros;
begin
  Inicia_Ents;
  Inicia_Atvs;
  Inicializa_Hist;
  Inicializa_SeriesT;
  Inicia_AD;
end;

procedure Inicio;
begin
  InicioVars;
  Set_Sist_Ent(SISTEMA);
  #
  Prepara_Termino(128,DURACAO);
  Cria_Registros;
  Atraves_C_Eventos;
end;

procedure Relatorio;
begin
# Open_File_Relatorio('NOMEARQ.REP');
  Valores_Atvs;
  Valores_Ents;
  Valores_Filas;
  Estatisticas_e_Histogramas;
  Series_Temporais;
# Close_File_Relatorio('NOMEARQ.REP');
end;
#

procedure Fim_Aquec;
begin
  Cria_Registros;
```

```
  Reatualiza_Ent(Arv_Tempo);
end;

function Fase_A:boolean;
begin
  Aumenta_Tempo;
  Fase_A := (TEMP <= Duracao);
end;

procedure Chama_B_Eventos;
begin
  while ( Tempo_Corrente(TEMP) ) do
  begin
    while (Prox_B_Evento) do
    begin
      case Nr_ProxB of
#
        127 : Fim_Aquec;
        128 : Fim_Corrida;
      end;
    end;
  end;
end;

procedure Atraves_C_Eventos;
begin
#
end;

procedure Executa;
begin
# Open_File('NOMEARQ.RST');
  Inicio;
  if ( Dur_Aquecimento <> 0 )
    then Prepara_Reinicio(127,Dur_Aquecimento);
  while (Fase_A) do
  begin
    Chama_B_Eventos;
    Atraves_C_Eventos;
  end;
  Relatorio;
# Close_File('NOMEARQ.RST');
end;

(* programa principal *)
begin
  Inicializa;
  Define_Modelo;
  repeat
    Define_Dados;
    if ( Duracao > 0 )  then Executa;
  until ( Duracao <= 0 );
  clrscr;
end.
#
```

- ANEXO 6 -

ARQUIVO BAR.PAS

```
(** Programa de Simulacao em Pascal 5.0 gerado pelo GERSIMUL. **)

program BAR;
Uses
  Crt,
  Simul;

var
  PORTA,SISTEMA,COPO,CLIENTE,GARCOM : Entidade;
  SUJO,FECHADA,ESPERA,LIMPO,CHEIO,PRONTO,OCIOSO : Fila;
  CHEGADA,SERVIR,BEBER,LAVAR : Atividade;
  RUA : Fonte;

procedure Define_Modelo;
begin
  Define_Entidade(SISTEMA,'SISTEMA');
  Define_Entidade(COPO,'COPO');
  Define_Entidade(CLIENTE,'CLIENTE');
  Define_Atributo(CLIENTE,'SEDE');
  Define_Entidade(GARCOM,'GARCOM');
  Define_Entidade(PORTA,'PORTA');
  Define_Hist_Atrib(CLIENTE,'SEDE','ATRIB_SEDE',1,0);
  Define_Fonte(CLIENTE,RUA,'RUA');
  Define_Fila(COPO,SUJO,'SUJO');
  Define_Fila(PORTA,FECHADA,'FECHADA');
  Define_Fila(CLIENTE,ESPERA,'ESPERA');
  Define_Hist_Fila(ESPERA,'F_ESPERA',Tamanho,1,0);
  Define_Hist_Fila(ESPERA,'T_ESPERA',Tempo,5,10);
  Define_Fila(COPO,LIMPO,'LIMPO');
  Define_Hist_Fila(LIMPO,'F_LIMPO',QLength,5,0);
  Define_Fila(COPO,CHEIO,'CHEIO');
  Define_Fila(CLIENTE,PRONTO,'PRONTO');
  Define_Fila(GARCOM,OCIOSO,'OCIOSO');
  Define_Atividade(CHEGADA,'CHEGADA');
  Define_Atividade(SERVIR,'SERVIR');
  Define_Atividade(BEBER,'BEBER');
  Define_Atividade(LAVAR,'LAVAR');
end;

procedure Atraves_C_Eventos;forward;

procedure Cria_Registros;
begin
  Inicia_Ents;
  Inicia_Atvs;
  Inicializa_Hist;
  Inicializa_SeriesT;
  Inicia_AD;
end;

procedure Inicio;
```

```
begin
  InicioVars;
  Set_Sist_Ent(SISTEMA);
  Chama_Enche_Fila('BAR.QUE');
  Prepara_Termino(128,DURACAO);
  Cria_Registros;
  Atraves_C_Eventos;
end;

procedure Relatorio;
begin
  Open_File_Relatorio('BAR.REP');
  Valores_Atvs;
  Valores_Ents;
  Valores_Filas;
  Estatisticas_e_Histogramas;
  Series_Temporais;
  Close_File_Relatorio('BAR.REP');
end;

procedure C_CHEGADA;
var cont:integer;
begin
  while ( Tam_Fila(FECHADA) >= 1)
  do
  begin
    Tempo_Atv:=NEGEXP(10,2);
    for cont:=1 to 1 do
    begin
      Def_Val_Atrib(Enesima_na_Fonte(RUA,cont),
      'SEDE',INTEIRA_UNIFORME(1,4,11);
      Acumula_Dados_Atrib(Enesima_na_Fonte(RUA,cont),
      'SEDE','ATRIB_SEDE');
    end;
    Programa_Atv(CHEGADA);
    PreparaB(1,Tira_da_Fila(frente,FECHADA));
    PreparaB(2,Tira_da_Fonte(RUA));
    Fim_Programa_Atv;
  end;
end;

procedure B1_Fim_CHEGADA_PORTA;
begin
  Coloca_na_Fila(Ent_corrente,Atras,FECHADA);
end;

procedure B2_Fim_CHEGADA_CLIENTE;
begin
  Acumula_Dados_Fila(ESPERA,'F_ESPERA');
  Coloca_na_Fila(Ent_corrente,Atras,ESPERA);
end;

procedure C_SERVIR;
begin
  while ( Tam_Fila(ESPERA) >= 1)
```

```
  and ( Tam_Fila(OCIOSO) >= 1)
  and ( Tam_Fila(LIMPO) >= 1)
  do
  begin
    Tempo_Atv:=NORMAL(6,1,5);
    Programa_Atv(SERVIR);
    Acumula_Dados_Fila(ESPERA,'F_ESPERA');
    PreparaB(3,Tira_da_Fila(frente,ESPERA));
    PreparaB(4,Tira_da_Fila(frente,OCIOSO));
    Acumula_Dados_Fila(LIMPO,'F_LIMPO');
    PreparaB(5,Tira_da_Fila(frente,LIMPO));
    Fim_Programa_Atv;
  end;
end;

procedure B3_Fim_SERVIR_CLIENTE;
begin
  Coloca_na_Fila(Ent_corrente,Atras,PRONTO);
end;

procedure B4_Fim_SERVIR_GARCON;
begin
  Coloca_na_Fila(Ent_corrente,Atras,OCIOSO);
end;

procedure B5_Fim_SERVIR_COPO;
begin
  Coloca_na_Fila(Ent_corrente,Atras,CHEIO);
end;

procedure C_BEBER;
begin
  while ( Tam_Fila(CHEIO) >= 1)
  and ( Tam_Fila(PRONTO) >= 1)
  do
  begin
    Tempo_Atv:=INTEIRA_UNIFORME(5,8,7);
    Def_Val_Atrib(Enesima_Ent(PRONTO,1),
    'SEDE',Avalia_Atrib(Enesima_Ent(PRONTO,1),'SEDE')-1);
    Programa_Atv(BEBER);
    PreparaB(6,Tira_da_Fila(frente,CHEIO));
    PreparaB(7,Tira_da_Fila(frente,PRONTO));
    Fim_Programa_Atv;
  end;
end;

procedure B6_Fim_BEBER_COPO;
begin
  Coloca_na_Fila(Ent_corrente,Atras,SUJO);
end;

procedure B7_Fim_BEBER_CLIENTE;
begin
  if( Avalia_Atrib(Ent_corrente,'SEDE')>0) then
  begin
```

```
    Acumula_Dados_Fila(ESPERA,'F_ESPERA');
    Coloca_na_Fila(Ent_corrente,Frente,ESPERA);
  end else
  begin
    Destroi(Ent_corrente);
  end;
end;

procedure C_LAVAR;
begin
  while ( Tam_Fila(SUJO) >= 1)
  and ( Tam_Fila(OCIOSO) >= 1)
  do
  begin
    Tempo_Atv:=5;
    Programa_Atv(LAVAR);
    PreparaB(8,Tira_da_Fila(frente,SUJO));
    PreparaB(9,Tira_da_Fila(frente,OCIOSO));
    Fim_Programa_Atv;
  end;
end;

procedure B8_Fim_LAVAR_COPO;
begin
  Acumula_Dados_Fila(LIMPO,'F_LIMPO');
  Coloca_na_Fila(Ent_corrente,Atras,LIMPO);
end;

procedure B9_Fim_LAVAR_GARCON;
begin
  Coloca_na_Fila(Ent_corrente,Atras,OCIOSO);
end;

procedure Fim_Aquec;
begin
  Cria_Registros;
  Reatualiza_Ent(Arv_Tempo);
end;

function Fase_A:boolean;
begin
  Aumenta_Tempo;
  Fase_A := (TEMP <= Duracao);
end;

procedure Chama_B_Eventos;
begin
  while ( Tempo_Corrente(TEMP) ) do
  begin
    while (Prox_B_Evento) do
    begin
      case Nr_ProxB of
          1 : B1_Fim_CHEGADA_PORTA;
          2 : B2_Fim_CHEGADA_CLIENTE;
          3 : B3_Fim_SERVIR_CLIENTE;
```

```
            4 : B4_Fim_SERVIR_GARCON;
            5 : B5_Fim_SERVIR_COPO;
            6 : B6_Fim_BEBER_COPO;
            7 : B7_Fim_BEBER_CLIENTE;
            8 : B8_Fim_LAVAR_COPO;
            9 : B9_Fim_LAVAR_GARCON;
          127 : Fim_Aquec;
          128 : Fim_Corrida;
        end;
      end;
    end;
end;

procedure Atraves_C_Eventos;
begin
  C_CHEGADA;
  C_SERVIR;
  C_BEBER;
  C_LAVAR;
end;

procedure Executa;
begin
  Open_File('BAR.RST');
  Inicio;
  if ( Dur_Aquecimento <> 0 )
    then Prepara_Reinicio(127,Dur_Aquecimento);
  while (Fase_A) do
  begin
    Chama_B_Eventos;
    Atraves_C_Eventos;
  end;
  Relatorio;
  Close_File('BAR.RST');
end;

(* programa principal *)
begin
  Inicializa;
  Define_Modelo;
  repeat
    Define_Dados;
    if ( Duracao > 0 ) then Executa;
  until ( Duracao <= 0 );
  clrscr;
end.
```

- ANEXO 7 -

ARQUIVO BAR.QUE

```
FECHADA
1   1
LIMPO
1   70
OCIOSO
1   3
==========
```