CELSO YUKIO EIZAK
VIVIANE MITIE HAMAMURA

Nota final
8,5 (oito e cinco)
22/01/04

SSH'03:
SIMULADOR DE SISTEMAS HÍBRIDOS

Trabalho de Formatura
apresentado à Escola Politécnica
da Universidade de São Paulo

São Paulo
2003

CELSO YUKIO EIZAK
VIVIANE MITIE HAMAMURA

SSH'03:
SIMULADOR DE SISTEMAS HÍBRIDOS

Trabalho de Formatura
apresentado à Escola Politécnica
da Universidade de São Paulo

São Paulo
2003

CELSO YUKIO EIZAK
VIVIANE MITIE HAMAMURA

SSH'03:
SIMULADOR DE SISTEMAS HÍBRIDOS

Trabalho de Formatura
apresentado à Escola Politécnica da
Universidade de São Paulo

Engenharia Mecatrônica

Orientador:
Prof. Dr. Paulo E. Miyagi
Co-orientador:
M. Eng. Emília Villani

São Paulo
2003

Aos meus pais e meus irmãos, que sempre estiveram do meu lado me apoiando e torcendo por mim em todas as minhas decisões.

À Sayuri pela companhia em todos os momentos.

À Viviane pela amizade, compreensão e dedicação.

Celso Yukio Eizak

Aos meus pais e meu irmão, pelo apoio, compreensão e carinho em todos os momentos.

Às minhas amigas do Cooper, pelo apoio e pelos momentos de descontração.

Ao Celso, pelo companheirismo, amizade e dedicação.

Viviane Mitie Hamamura

AGRADECIMENTOS

Ao orientador Prof. Dr. Paulo Eigi Miyagi,

por todo conhecimentos e experiência transmitidos em nossos encontros.

À co-orientadora Emília Villani,

pela grande ajuda e atenção nos momentos difíceis durante todo o período de desenvolvimento do trabalho.

Aos amigos da Poli

que estiveram juntos, tornando esses anos na faculdade mais agradável.

À todos

que de alguma forma contribuíram com nossa formação.

RESUMO

O presente trabalho visa desenvolver um Simulador de Sistemas Híbridos baseado na representação destes sistemas em Rede Predicado Transição Diferencial. O trabalho envolveu uma pesquisa bibliográfica sobre Sistemas a Eventos Discretos (SED) e Sistemas de Variáveis Contínuas (SVC) e métodos de resolução de equações diferenciais.

Este projeto também considerou aspectos funcionais do simulador de modo a oferecer interface gráfica amigável ao usuário e o menor tempo computacional possível.

São evidenciados os problemas encontrados durante o desenvolvimento do simulador bem como as soluções propostas para os mesmos.

Além disso o trabalho traz um exemplo de aplicação do Simulador, deixando claro a importância do seu desenvolvimento e a aplicabilidade em processos industriais.

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

Entre as diversas atividades do projeto de sistemas produtivos, a construção de um modelo e sua análise estão entre as mais importantes. Através destas atividades é possível estudar o comportamento do sistema produtivo e detectar erros e falhas antes mesmo de sua implementação.

No que se refere à modelagem, os sistemas produtivos podem ser classificados em Sistemas a Eventos Discretos (SED), Sistemas de Variáveis Contínuas (SVC) e Sistemas Híbridos. [Antsaklis, 1998].

Em um SED, o estado do sistema é alterado pela ocorrência de eventos considerados instantâneos, isto é, onde o tempo de duração do evento não é relevante em relação ao tempo de permanência dos estados. A ocorrência destes eventos depende apenas da satisfação de pré e pós-condições do sistema [Miyagi, 1996]. Por outro lado em um SVC a evolução dos estados do sistema é função do tempo [Villani, 2000], isto é, o estado do sistema é representado por variáveis que são modificadas de forma contínua no tempo assumindo infinitos valores. Os Sistemas Híbridos por sua vez são definidos como aqueles onde estão presentes características tanto de SED quantos de SVC.

Segundo Villani [2000] poucos processos reais, quando analisados sobre o ponto de vista de integração com outros sistemas, podem ser considerados inteiramente contínuos ou discretos. A maioria dos processos contínuos sofre modificações discretas significantes que se sobrepõem ao seu comportamento dinâmico. Exemplos típicos dessas interferências são falhas em equipamentos, modificações planejadas, início e conclusão de operação, operações de manutenção, etc. Situações semelhantes existem em sistemas discretos, onde a ocorrência de eventos é influenciada pela dinâmica da parte contínua. Sendo assim, em muitos casos nenhuma das partes, de eventos discretos e de dinâmica contínua, podem ser desconsideradas ou omitidas. Portanto, é desejável que a modelagem, análise e controle do sistema sejam desenvolvidos de forma híbrida.

1.1 Motivação

Entre as atividades do projeto de sistemas de controle para sistemas produtivos, está a construção de um modelo e sua análise. Este modelo deve não apenas representar o comportamento do sistema de controle, mas também do seu objeto de controle. O objetivo da análise é então verificar se o sistema de controle impõe o comportamento desejado ao sistema produtivo. Existem duas formas principais de se analisar um modelo: verificação formal de propriedades e simulação. Enquanto a verificação formal de propriedades pode ser feita de forma automática ou manual, a simulação em geral é realizada de forma automática por uma ferramenta computacional.

De uma forma geral a simulação sempre é usada para uma primeira análise e detecção de erros no modelo construído [Silva, 2000], uma vez que sua aplicação é mais simples e rápida.

A importância da simulação é ressaltada pelo fato de que, em muitos casos, ela é a única solução viável, pois a verificação de propriedades usando formalismos matemáticos nem sempre é possível, principalmente para sistemas complexos e de grande porte. De acordo com Astrom [1998], "formalismos matemáticos são freqüentemente inadequados para muitos problemas que urgem de uma solução, e, na ausência de evoluções radicais na matemática, a simulação é a mais promissora e poderosa forma de se atacar o problema... Este é geralmente o caso dos problemas de engenharia".

Para sistemas híbridos a situação é particularmente crítica, pois as ferramentas e métodos existentes são capazes apenas de resolver problemas relativamente simples [Silva, 2001].

1.2 Objetivos

O objetivo deste trabalho é desenvolver um simulador de sistemas híbridos. Esse simulador deve permitir a edição e simulação de modelos de sistemas que tenham uma parte puramente discreta, uma parte puramente contínua e uma parte híbrida que representa a interface entre a parte contínua e a parte discreta.

Para modelar a parte discreta do sistema, considerou-se as redes de Petri devido as suas bem conhecidas características para descrição de paralelismo, concorrência, sincronismo, etc, entre processos, além da simplicidade de representação gráfica aliada a uma poderosa representação matemática.[Cardoso, 1997]. Para modelar a parte contínua são utilizados sistemas de equações diferenciais. Para a parte híbrida são utilizadas as redes Predicado Transição Diferenciais (redes PTD), que associam redes de Petri a sistemas de equações diferenciais. [Champagnat, 1998].

O desenvolvimento de um simulador para sistemas híbridos baseado em redes PTD é motivado pelo fato de que os simuladores já propostos nesta área apresentam os seguintes problemas:

- A parte discreta deve ser especificada usando autômatos e portanto a modelagem de paralelismos no sistema fica comprometida (ex: [Henzinger, 1997]).
- O simulador se baseia nas redes de Petri Híbridas, onde o poder de modelagem da parte contínua é bastante restrito (ex: [Drath, 1998]).

2 REVISÃO BIBLIOGRÁFICA

2.1 Ferramentas de modelagem

Um processo pode ser classificado como um Sistema Híbrido, se esse processo apresentar características tanto de SED quanto de SVC.

Além disso em função das diferentes características envolvidas, para cada tipo de sistema é utilizada uma ferramenta de modelagem diferente. No caso de SED, existem diversos trabalhos que utilizam as Redes de Petri. Para SVC, os Sistemas de Equações Diferenciais são as técnicas consagradas e amplamente adotadas. Para Sistemas Híbridos, este trabalho considera as Redes de Predicado Transição Diferenciais conforme é posteriormente justificado. Cada uma dessas ferramentas de modelagem são descritas nos itens a seguir.

2.1.1 Modelagem de SED - Redes de Petri

Rede de Petri é uma técnica poderosa para o estudo, análise e modelagem de sistemas a eventos discretos (SED). Ela pode ser representada de forma gráfica (grafos) ou na forma de matrizes [Maciel, 1996].

A representação gráfica da rede de Petri é baseada em grafo bipartido e orientado, isto é, são considerados dois tipos de nós, os lugares (*places* – representados por círculos) e as transições (*transitions* – representados por barras ou retângulos). A ligação entre os nós é feita por arcos orientados (setas). A Figura 1 abaixo mostra os elementos de uma rede de Petri [Maciel, 1996]:

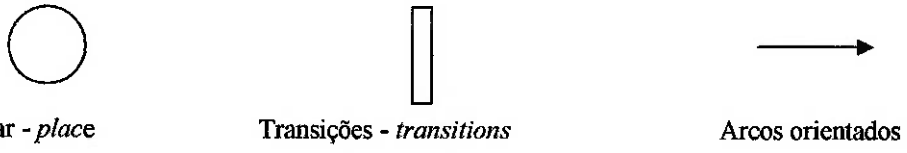

Figura 1 – Elementos de uma Rede de Petri.

A rede de Petri é um multigrafo, isto é, ela permite múltiplos arcos de um nó do grafo para outro, no entanto estas ligações devem respeitar uma condição: os arcos orientados não podem ligar diretamente lugares a lugares ou ligar transições a transições [Maciel, 1996]. Esta regra está ilustrada nas

Figura 2 e Figura 3.

Figura 2 - Exemplos de construções inconsistentes.

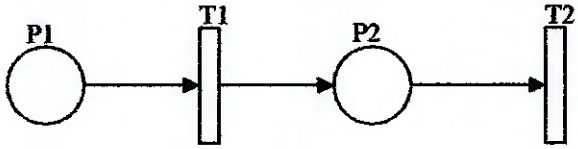

Figura 3 – Exemplo de modo consistente de rede de Petri.

A rede de Petri é marcada quando há marcas no interior dos lugares. Cada marcação corresponde a um determinado estado do sistema modelado. No exemplo apresentado na Figura 4, a rede de Petri é uma rede marcada com marcas nos lugares P1 e P2.

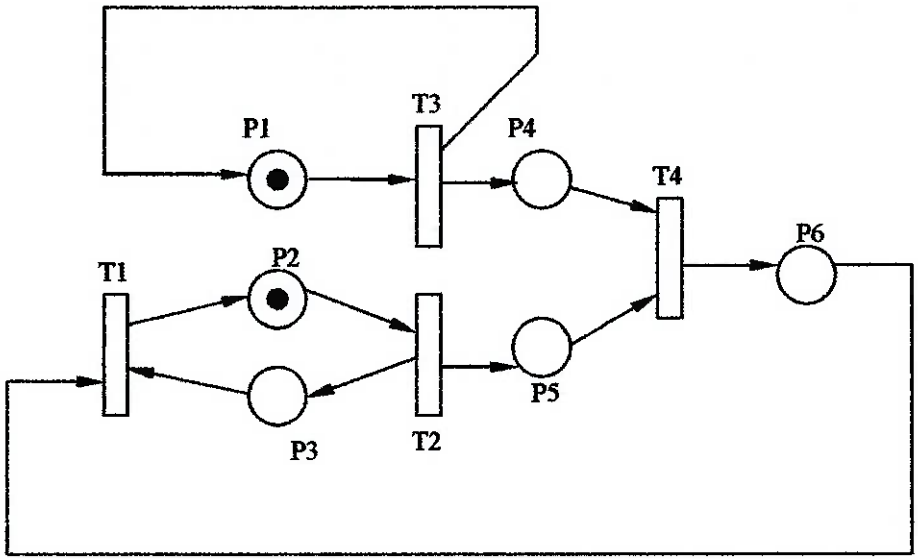

Figura 4 – Exemplo de rede de Petri

Em relação às mesmas marcas e marcações, existem diferentes tipos de redes de Petri, cujos principais são:

- Redes de Petri Condição/Evento – neste tipo de rede cada lugar pode receber no máximo uma marca.
- Redes de Petri Lugar/Transição – ao contrário das redes condição/evento, os lugares podem conter mais de uma marca. É possível ainda determinar pesos para os arcos e capacidade (número máximo de marcas) para os lugares.
- Redes de Petri de Alto Nível – as marcas são individualizadas, isto é existem tipos de marcas diferentes no grafo.

2.1.1.1 Regras de execução

A execução da rede de Petri é baseada no número e distribuição das marcas ao longo dos lugares da rede. A rede de Petri é executada através dos *disparos* das transições. As transições só podem ser disparadas quando estão *habilitadas*.

Diz-se que um elemento A está à entrada do elemento B quando existe um arco orientado que tem a origem no elemento A e destino no elemento B, e diz-se que um elemento A está à saída do elemento B quando o arco orientado tem o destino no elemento A e a origem no elemento B.

Nas redes condição/evento, uma transição está habilitada quando todos os lugares que estão à sua entrada estão marcados e todos lugares à sua saída não estão marcados.

No exemplo ilustrado pela Figura 5 tem-se uma rede de Petri do tipo Condição/Evento e pode-se notar que os lugares à entrada da transição T1 estão marcados e o lugar à saída está livre, portanto a transição T1 está habilitada.

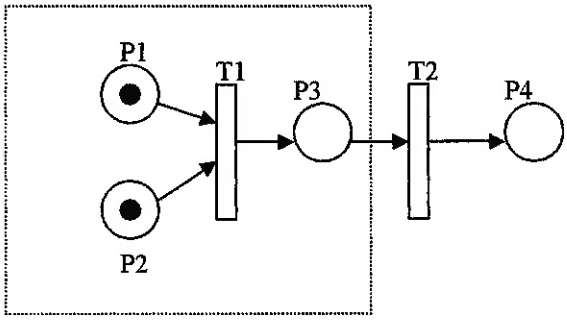

Figura 5 – Transição T1 habilitada, antes do disparo

A transição estando habilitada, dispara. Ao ocorrer o disparo, as marcas dos lugares que estão à entrada da transição são retiradas e novas marcas são colocadas

nos lugares à saída da transição No exemplo anterior após o disparo da T1 a rede fica com a configuração da Figura 6.

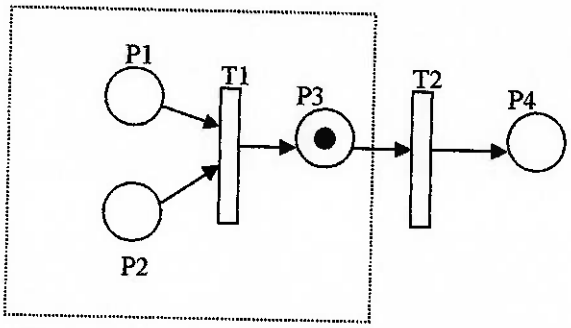

Figura 6 – Marcação obtida após o disparo de transição T1

Para redes Lugar/Transição o lugar de saída não necessita estar livre, basta que o lugar à saída tenha uma capacidade suficiente para receber as marcas adicionais. O número de marcas a serem retiradas e adicionadas aos lugares é definido pelo peso dos respectivos arcos. Caso os arcos de entrada da rede Lugar/Transição possuam um peso n, então se deve retirar n marcas dos respectivos lugares à entrada da transição que está sendo disparada. E no caso dos arcos de saída da rede Lugar/Transição terem um peso m, então se deve acrescentar m marcas nos lugares respectivos à saída da transição. Para ilustrar a regra citada apresenta-se a Figura 7 e a Figura 8.

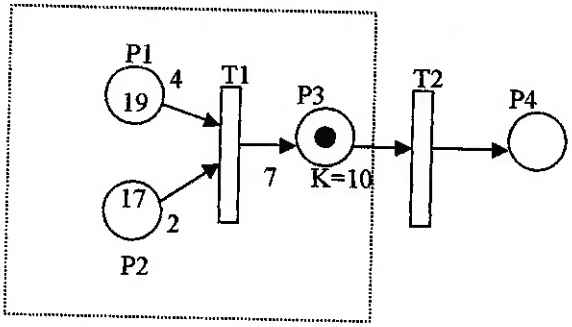

Figura 7 – Exemplo de rede de Petri Lugar-Transição com peso nos arcos

Após o disparo do T1 na Figura 7, a rede fica com a nova marcação mostrada na Figura 8.

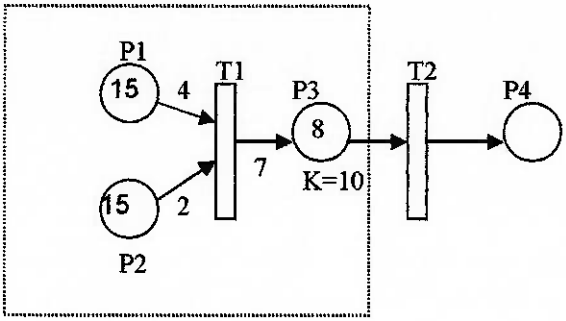

Figura 8 – Marcação da rede da Figura 7 após o disparo da transição T1

2.1.2 Modelagem de SVC – Sistemas de Equações Diferenciais

Os SVC envolvem estados e eventos que variam continuamente no tempo. As variáveis de estados e a eventos tratados como variáveis de controle, entradas, variáveis do objeto de controle e saída do sistema mantém uma relação que pode ser descrita (ou pelo menos aproximada) por um sistema de equações diferenciais ordinárias. Evidentemente existem sistemas que podem envolver sistemas de equações de outra natureza, entretanto, uma grande parte dos casos práticos têm sido tratados com sistemas de equações diferenciais ordinárias.

Segundo [Rice, 1993], uma equação diferencial ordinária tem a forma geral:

$$F\left(x, y, \frac{dy}{dx}, \frac{d^2y}{dx^2}, \frac{d^3y}{dx^3}, \ldots, \frac{d^ny}{dx^n}\right) = 0 \qquad (1)$$

A equação apresentada é chamada de equação ordinária de n-ésima ordem. Ela é uma equação ordinária porque há somente uma variável independente, x. É de n-ésima ordem porque a maior derivada é de ordem n.

Uma função $y(x)$, n vezes diferenciável, satisfazendo a equação anterior é chamada de solução desta equação. As equações diferenciais ordinárias têm várias soluções. É necessário que sejam dadas informações adicionais sobre $y(x)$ e/ou sobre suas derivadas em valores específicos de x para que ela seja a solução única. Para uma equação diferencial de ordem n, normalmente são suficientes n condições adicionais para garantir que a solução $y(x)$ seja única. Se todas as n condições adicionais forem especificadas para um mesmo valor de x, x_0 por exemplo, temos um problema conhecido como Problema do Valor Inicial, PVI. Caso estas n

condições adicionais sejam dadas para mais de um valor de x, temos um problema conhecido como Problema de Valor de Contorno, PVC.

Em geral, é relativamente difícil a obtenção de soluções analíticas para equações diferenciais. Na maioria dos casos as soluções devem ser geradas através de métodos numéricos. Alguns métodos utilizados são: Método de Euler, Método de Heun, Método de Butcher e Método de Runge-Kutta.

2.1.2.1 Comparação entre os métodos numéricos

Através da análise comparativa do tempo computacional envolvido e os erros gerados nos métodos considerados (método de Euler [Euler], método de Heun [Heun], série de Taylor e Runge-Kutta [RK3d= terceira ordem, RK4th= quarta ordem]) verificou-se que o método de Runge-Kutta de 4ª ordem apresenta melhor relação 'custo computacional/erro' em relação aos outros métodos, como podemos verificar na Figura 9, [Rice, 1993].

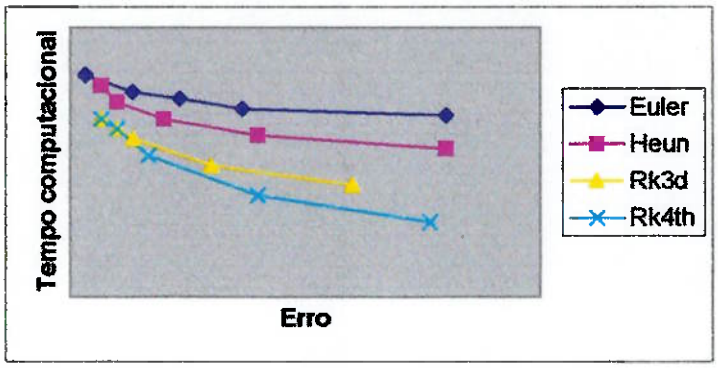

Figura 9 - Comparação entre métodos numéricos

Assim, para sistemas de variáveis continuas, especificamente para resolução de sistemas de equação diferencial optou-se pela utilização neste trabalho do método de Runge-Kutta de quarta ordem.

2.1.2.2 Método de Runge-Kutta de 4ª ordem

O método de Runge-Kutta de 4ª ordem utiliza valores de $f(x,y)$ no intervalo $[x_n, x_n + h]$ para obter uma "boa" aproximação para y_{n+1}.

$$y_{k+1} = y_k + \frac{h}{6}[f_1 + 2f_2 + 2f_3 + f_4] \qquad (2)$$

com os coeficientes f dados por:

$$f_1 = f(x_k, y_k) \tag{3}$$

$$f_2 = f\left(x_k + \frac{h}{2}, y_k + \frac{h}{2} f_1\right) \tag{4}$$

$$f_3 = f\left(x_k + \frac{h}{2}, y_k + \frac{h}{2} f_2\right) \tag{5}$$

$$f_4 = f(x_k + h, y_k + h f_3) \tag{6}$$

2.1.2.3 Sistema de equações diferenciais

Considerando um sistema de equações diferenciais ordinárias de 1ª ordem:

$$\frac{du}{dx} = f(x,u,v) \qquad \text{com} \qquad u(x_0) = u_0 \tag{7}$$

$$\frac{dv}{dx} = g(x,u,v) \qquad \text{com} \qquad v(x_0) = v_0 \tag{8}$$

Pelo método de Runge-Kutta de 4ª ordem, as aproximações para as soluções das equações são geradas através das ralações:

$$u_{k+1} = u_k + \frac{h}{6}(f_1 + 2f_2 + 2f_3 + f_4) \tag{9}$$

e

$$v_{k+1} = v_k + \frac{h}{6}(g_1 + 2g_2 + 2g_3 + g_4) \tag{10}$$

com :

$$f_1 = f(x_k, u_k, v_k) \tag{11}$$

$$f_2 = f\left(x_k + \frac{h}{2}, u_k + \frac{h}{2} f_1, v_k + \frac{h}{2} g_1\right) \tag{12}$$

$$f_3 = f\left(x_k + \frac{h}{2}, u_k + \frac{h}{2} f_2, v_k + \frac{h}{2} g_2\right) \tag{13}$$

$$f_4 = f(x_k + h, u_k + h f_3, v_k + h g_3) \tag{14}$$

e

$$g_1 = g(x_k, u_k, v_k) \tag{15}$$

$$g_2 = f\left(x_k + \frac{h}{2}, u_k + \frac{h}{2} f_1, v_k + \frac{h}{2} g_1\right) \tag{16}$$

$$g_3 = f\left(x_k + \frac{h}{2}, u_k + \frac{h}{2} f_2, v_k + \frac{h}{2} g_2\right) \tag{17}$$

$$g_4 = f\left(x_k + h, u_k + h f_3, v_k + h g_3\right) \tag{18}$$

2.1.3 Modelagem de Sistemas Híbridos - Redes Predicado/ Transição Diferencial (PTD)

Este tipo de Rede de Petri concebido para a modelagem de sistemas híbridos diferencia-se da Rede de Petri Condição/Evento nos seguintes aspectos:

a) Um vetor de variáveis é associado à rede.

b) Os lugares representam configurações do sistema e são associados a sistemas de equações, que envolvem as variáveis citadas em a).

c) As transições possuem "funções de habilitação" e "funções de junção", que envolvem variáveis citadas em a).

Quando um lugar está marcado, o sistema de equações a ele associado determina a evolução contínua no tempo das variáveis.

Para o disparo das transições, os valores das variáveis contínuas que são responsáveis pela sua habilitação são fornecidos como parâmetros de entrada de uma função de habilitação do tipo: "se a expressão 1 for maior, igual ou menor que a expressão 2 então a transição está habilitada". É preciso lembrar que neste tipo de rede o disparo segue também todas as regras da rede de Petri Condição/Evento, porém é adicionada uma condição que depende das variáveis contínuas.

Ainda a respeito das transições na rede PTD, o segundo tipo de função é chamado de função de junção. Este tipo de função atualiza o valor das variáveis contínuas quando a transição é disparada, em outras palavras, ela modifica as variáveis contínuas segundo algumas regras pré-estabelecidas pelo modelo, de uma forma discreta.

A Figura 10 apresenta um exemplo de uma rede PTD.

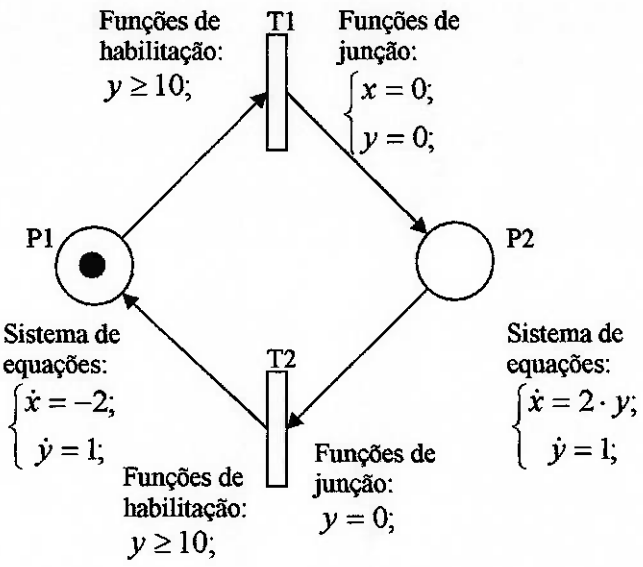

Figura 10 - Exemplo de rede PTD.

3 SIMULADOR DESENVOLVIDO

3.1 Escolha do ambiente de programação

Considerando os softwares existentes e as ferramentas de modelagem citadas anteriormente, as seguintes propostas foram estudadas como possibilidade para implementação de um simulador de sistemas híbridos:

a) **MatLab e Simulink** – Baseado num pacote amplamente utilizado por pesquisadores e profissionais na área de controle, o simulador seria implementado usando subrotinas do *software* Matlab para resolução da parte discreta (disparo das transições da rede de Petri) e a ferramenta Simulink para a resolução dos sistemas de equações diferenciais. A interface entre as partes contínua e discretas também seria realizada através do Simulink.

b) **MatLab e C++** - Visando melhorar o tempo de execução dos experimentos, o *software* Matlab seria utilizado para a simulação da parte discreta. Já a parte contínua seria implementada como subrotinas na linguagem de programação C++, eliminando a utilização do SIMULINK e objetivando aumentando a velocidade de simulação. A sincronização entre a parte contínua e discreta também seria implementada como subrotinas no MatLab.

c) **C++** - Desconsiderando agora qualquer pacote existente de análise de sistemas um simulador completamente novo seria confeccionado apenas na linguagem C++, assim todas as funções relacionadas com a simulação a eventos discretos, do sistema híbrido e também do sistema a variáveis contínuas devem ser desenvolvidas.

A opção descrita no item a) foi adotada por Yen-Tsang [2001], e apresentou como principais problemas o tempo elevado de simulação e a necessidade de que o usuário tenha um razoável conhecimento do pacote MatLab e Simulink. Devido a estes problemas esta opção foi desconsiderada. Entre as opções b) e c), a do item c) foi selecionada, pois assegura potencialmente um menor tempo de simulação.

O simulador de Sistemas Híbridos foi então implementado na linguagem de programação C++, utilizando o software Borland Builder C++ 4.

A implementação da parte discreta considerou alguma estruturas e funcionalidades do *software* Petri Net Simulator [Takada, 1999]. A implementação da parte contínua e híbrida foi realizada utilizando os conceitos apresentados anteriormente.

3.2 Especificação da estrutura do simulador

Seguindo a opção adotada por [Takada, 1999] na implementação do software Petri Net Simulator, adotou-se uma abordagem funcional, para decomposição e organização do simulador.

Do ponto de vista estrutural, o simulador é formado por um conjunto de funções organizadas de forma hierárquica conforme descrito a seguir.

As funções do simulador estão organizadas em 4 blocos principais (Figura 11). Cada bloco realiza um conjunto de tarefas do simulador e são detalhadas na seqüência.

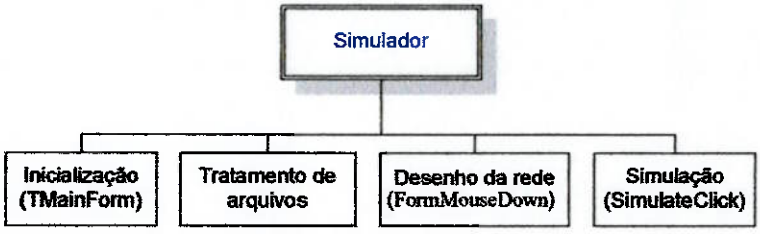

Figura 11 - Estrutura do simulador.

3.2.1 Inicialização (TMainForm)

Quando o programa é executado, ocorre a inicialização de todas as ferramentas necessárias para o funcionamento do *software*, bem como de todas as variáveis do programa. Esta inicialização é executada pela função TMainForm, que é composta das funções, indicadas na Figura 12:

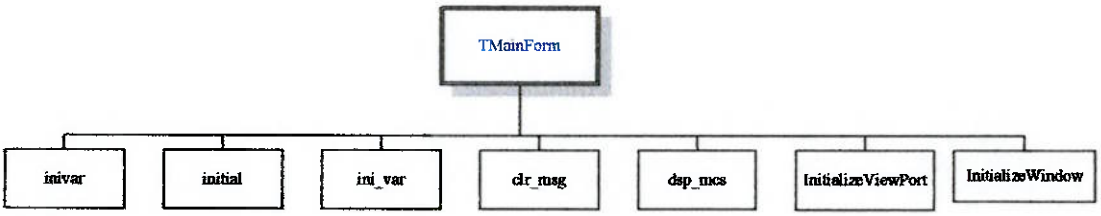

Figura 12 - Funções do bloco de Inicialização (TMainForm).

3.2.1.1 Função inivar

Realiza a inicialização das variáveis globais. Estas variáveis indicam, entre outras coisas, os números dos lugares, das transições e dos arcos, as marcações dos lugares, e ocorrências de erros no funcionamento do simulador.

3.2.1.2 Função initial

Realiza a inicialização das variáveis Fname (variável que guarda temporariamente o nome do arquivo quando o usuário digita) e Tname, relacionado com o titulo do arquivo, neste caso é o nome do arquivo quando é salvo.

3.2.1.3 Função ini_var

Inicializa alguns apontadores e libera memória.

3.2.1.4 Função clr_msg

Apaga eventuais mensagens de erro.

3.2.1.5 Função dsp_mcs

Habilita o mouse.

3.2.1.6 Função InitializeViewPort

Abre a janela de edição de modelos (o *ViewPort*).

3.2.1.7 Função InitializeWindow

Abre a janela principal do simulador.

3.2.2 Tratamento de Arquivos

As funções relativas ao tratamento de arquivos estão representadas na Figura 13.

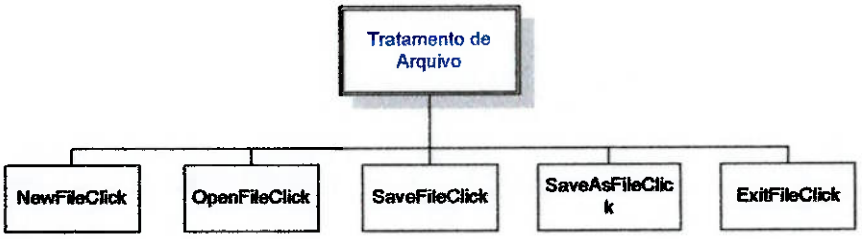

Figura 13 – Funções do bloco de Tratamento de Arquivos.

3.2.2.1 Função NewFileClick

Abre um novo arquivo.

3.2.2.2 Função OpenFileClick

Abre um arquivo de uma pasta ou diretório. A função OpenFileClick utiliza a função OpenXFile, e esta chama a função dat_lod que carrega o arquivo escolhido pelo usuário e, por sua vez, utiliza um conjunto de funções, conforme apresentado na Figura 14.

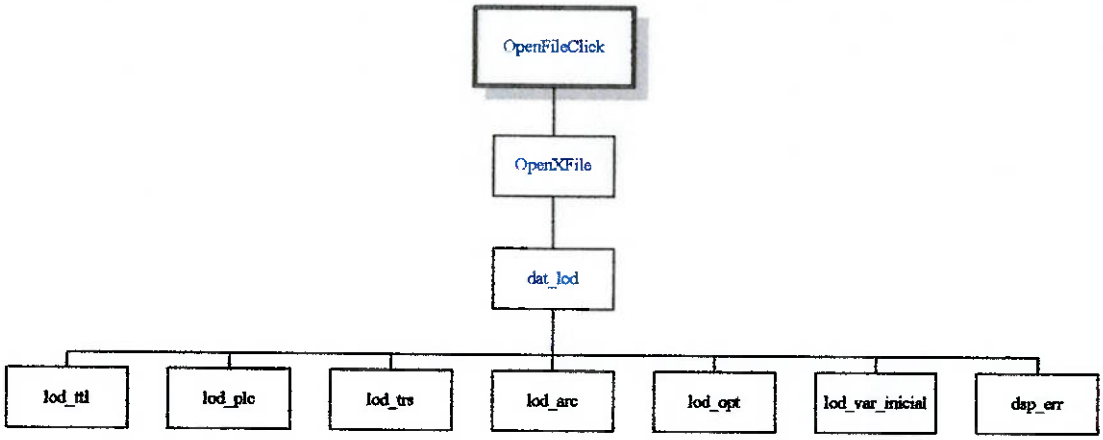

Figura 14 - Funções de OpenFileClick.

O conjunto de funções utilizado por dat_lod é composto por:

- lod_ttl: carrega o título do arquivo. Variáveis globais usadas: Tname.
- lod_plc: carrega os dados referentes aos lugares da rede no arquivo. Variáveis globais usadas: place, place2, plc_no, plc_no2.
- lod_trs: carrega os dados referentes às transições da rede no arquivo. Variáveis globais usadas: trs_no, trs_no2, trans, trans2.
- lod_arc: carrega dados do arco no arquivo. Variáveis globais usadas: arc_no, arc_no2, arcc, arcc2.
- lod_opt:carrega opções de simulação no arquivo.
- lod_var_inicial : carrega os valores iniciais das variáveis. Variáveis globais usadas: var_inicial, var_inicial_len.
- dsp_err: exibe mensagem de erro.

3.2.2.3 Função SaveFileClick

A função SaveFileClick salva os dados do modelo presente no ViewPort no arquivo que está aberto. Esta função utiliza outra função, dat_sav, que por sua vez utiliza um conjunto de funções conforme apresentado na Figura 15.

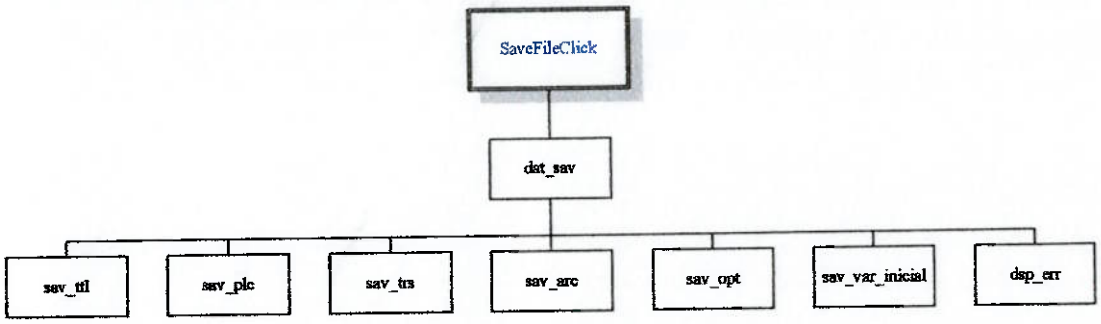

Figura 15 - Funções de SaveFileClick

- sav_ttl: salva o título do arquivo. Variáveis globais usadas: Tname.
- sav_plc: salva os dados referentes aos lugares no arquivo. Variáveis globais usadas: place, place2, plc_no, plc_no2.
- sav_trs: salva os dados referentes às transições no arquivo. Variáveis globais usadas: trs_no, trs_no2, trans, trans2.
- sav_arc: salva dados referentes aos arcos no arquivo. Variáveis globais usadas: arc_no, arc_no2, arcc, arcc2.
- sav_opt: salva opções de simulação no arquivo.
- sav_var_inicial: salva os valores iniciais das variáveis. Variáveis globais usadas: var_inicial, var_inicial_len.
- dsp_err: exibe mensagem de erro.

3.2.2.4 Função SaveAsFileClick

Esta função também chama a função dat_sav. Porém, o usuário escolhe outro local ou nome para o arquivo.

3.2.2.5 Função ExitProgramClick

A função ExitProgramClick chama a função fre_arq_all e a Close, como mostrado na Figura 16.

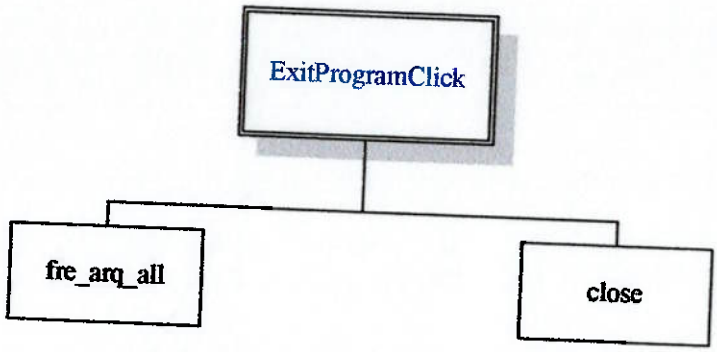

Figura 16 – Funções de ExitProgramClick

- fre_arq_all: libera memória de toda lista de arquivos. Variáveis globais usadas: arch.
- close: fecha o simulador.

3.2.3 Desenho da rede (FormMouseDown)

A edição da rede PTD é realizada pela função FormMouseDown, chamada quando o usuário aperta o botão esquerdo do mouse dentro da janela do ViewPort. Esta função lê as coordenadas x e y (caso o programa não esteja em simulação) e chama a função edt_png que por sua vez utiliza um conjunto de funções para edição da rede de acordo com as opções selecionadas pelo usuário.

A Figura 17 mostra esta estrutura.

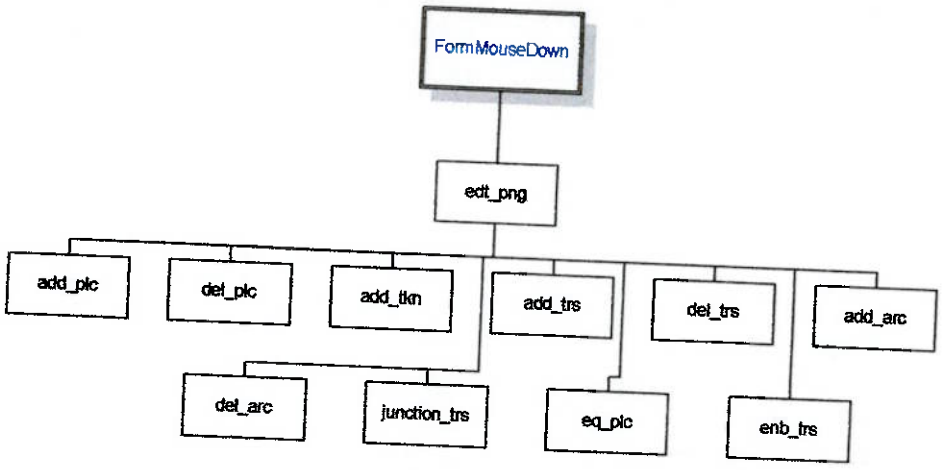

Figura 17 - Funções do bloco de Desenho da Rede.

3.2.3.1 Funções relacionadas à edição de lugares

- add_plc: insere um lugar na tela: Variáveis globais usadas: nome, plc_no (ou plc_no2).

- del_plc: apaga um lugar na tela: Variáveis globais usadas: arcc, plc_no (ou plc_no2).
- add_tkn: adiciona marca ao lugar selecionado.
- eq_plc: insere um sistema de equações no lugar selecionado.

3.2.3.2 Funções relacionadas à edição de transições

- add_trs: insere uma transição na tela. Variáveis globais usadas: trs_no (ou trs_no2), trs_ptn.
- del_trs: apaga uma transição da tela. Variáveis globais usadas: trs_no (ou trs_no2), trs_ptn.
- junction_trs: insere a função de junção na transição selecionada. Variáveis globais usadas: trs_no (ou trs_no2).
- enb_trs: insere a função de habilitação na transição selecionada. Variáveis globais usadas: trs_no (ou trs_no2).

3.2.3.3 Funções relacionadas à edição dos arcos

- add_arc: adiciona arco. Variáveis globais usadas: arc_ptn.
- del_arc: apaga arco. Variáveis globais usadas: arcc (ou arcc2), arc_no (ou arc_no2).

3.2.4 Simulação (Simulation)

Este bloco permite ao usuário definir as opções de simulação e iniciar a simulação após abrir um arquivo existente ou a construção da rede. Ao final da simulação é possível a visualização do comportamento das variáveis em função do tempo. A estrutura deste bloco é apresentada na Figura 18.

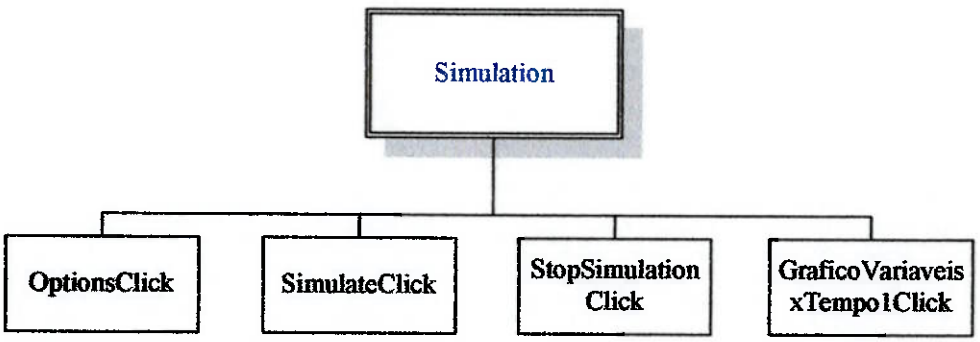

Figura 18 - Funções de Simulation.

3.2.4.1 Função OptionsClick

Esta sub-rotina abre uma caixa de diálogo para seleção de opções de simulação.

3.2.4.2 Função SimulateClick

Inicia a simulação da rede. Esta função chama a função TpeThread, que por sua vez utiliza as funções exe_man e exe_man2. A Figura 19 mostra esta estrutura.

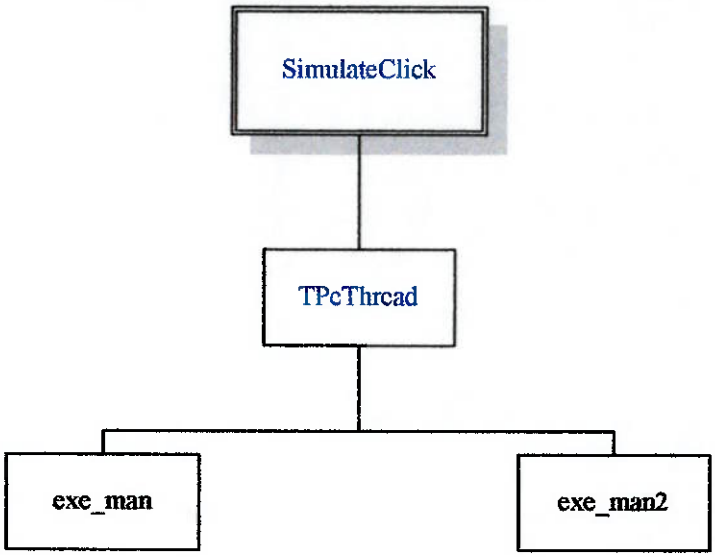

Figura 19 - Funções de SimulateClick.

3.2.4.2.1 Função exe_man

Esta função executa a simulação. A Figura 20 mostra a estrutura desta função.

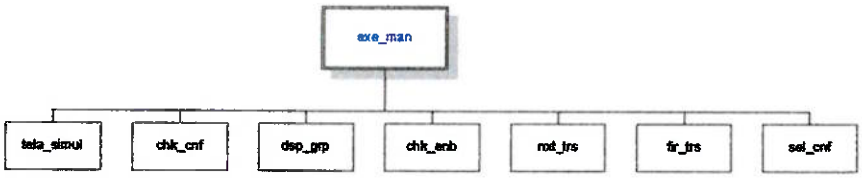

Figura 20 - Funções de exe_man.

3.2.4.2.1.1 Função tela_simul

Esta função atualiza a marcação da rede durante a simulação. Esta função trabalha com mais duas funções: clr_trg e dsp_grp. Esta estrutura é apresentada na Figura 21. A função clr_trg reinicia as variáveis globais do mouse. A função dsp_grp

desenha no *Viewport* todos os arcos, lugares e transições, usando um conjunto de funções, conforme a Figura 22.

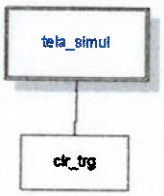

Figura 21 - Funções de tela_simul.

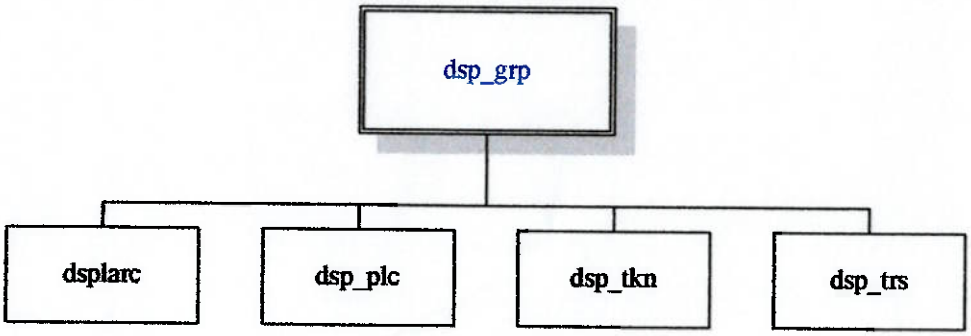

Figura 22 - Funções de dsp_grp.

3.2.4.2.1.2Função chk_enb

Esta função procura as transições que podem ser disparadas do ponto de vista discreto.

3.2.4.2.1.3Função nxt_trs

Esta função seleciona a próxima transição a ser disparada.

3.2.4.2.1.4Função fir_trs

Esta função dispara a transição que esta habilitada do ponto de vista discreto se a mesma também está habilitada do ponto de vista contínuo, e atualiza os valores das variáveis contínuas. A estrutura desta função está representada na Figura 23.

- A função chk_jfcn verifica se existe função de junção associada à transição. A função juncao_variavel atualiza o valor da variável.
- A função chk_efcn procura transições disparáveis do ponto de vista contínuo. A função enb_fcn verifica se a função de habilitação de uma transição está satisfeita. A função fir_eq calcula o sistema de equação associado ao lugar. A função rk4 calcula

o Método de Runge-Kutta de 4ª ordem. A função funcao calcula cada parcela para o método de Runge-Kutta. A função dae calcula as equações algébricas do sistema.

- A função at_var atualiza os valores das variáveis.
- A função set_fir retira a marca de um lugar e insere no próximo lugar.
- A função dsp_fir mostra a transição disparando, ou seja, quando a transição está habilitada ela fica com a cor verde. Depois que disparou ela volta à cor preta. Para isto são necessárias as funções: dsp_trs, que desenha a transição, e dsp_tkn, que desenha a marca.

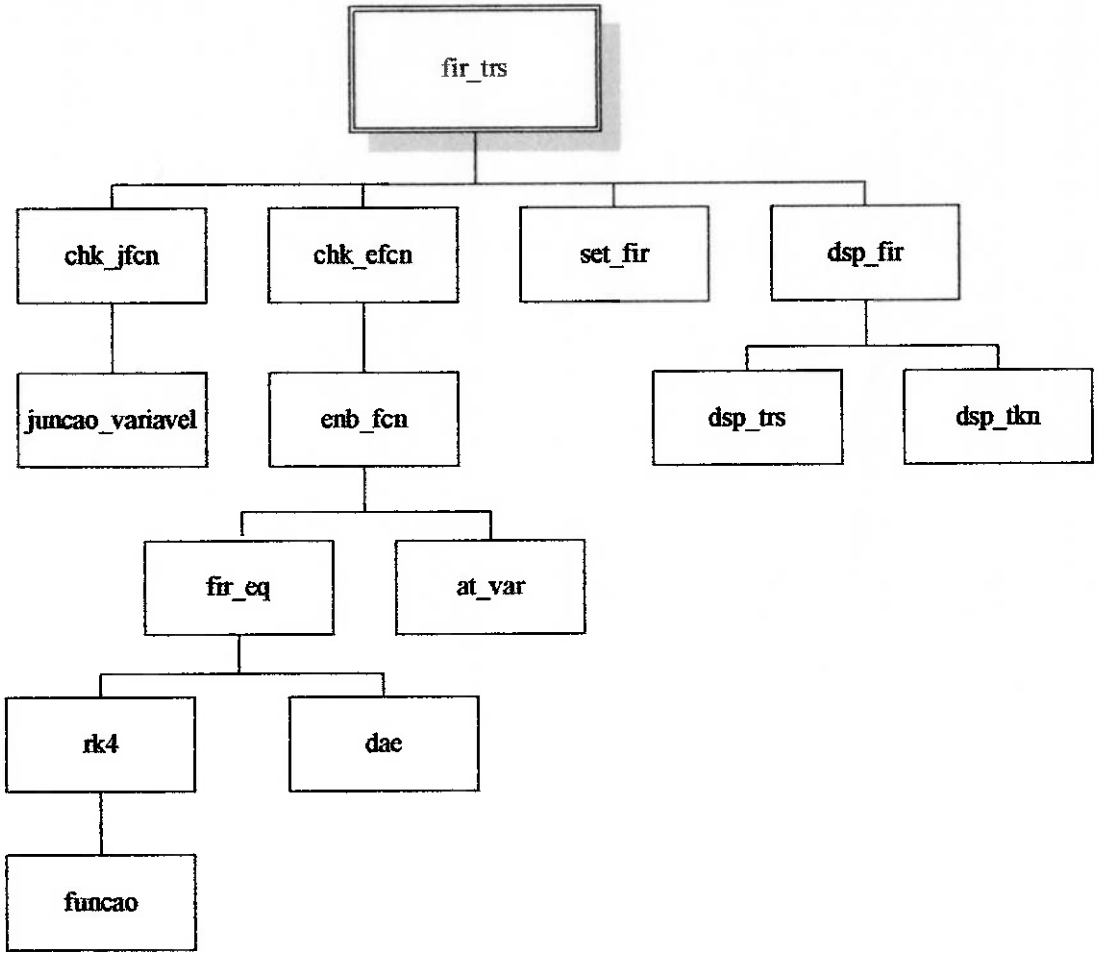

Figura 23 - Funções de fir_trs

3.2.4.2.1.5Função sel_cnf

Seleciona quais as transições disponíveis para ativar.

3.2.4.2.1.6Função chk_cnf4

Procura por transições em conflito com a transição habilitada.

3.2.4.2.2 Função exe_man2

Executa simulação com disparos simultâneos de transição. As funções utilizadas são as mesmas que a função exe_man.

3.2.4.3 Função StopSimulationClick

A função StopSimuationClick pára a simulação quando o tempo máximo de simulação é atingido.

3.2.4.4 Função GraficoVariaveisxTempo1Click

Esta função mostra na forma de gráfico a evolução do valor das variáveis escolhidas em um intervalo de tempo determinado pelo usuário.

3.3 Especificação dinâmica do simulador

O funcionamento interno do simulador é descrito através dos fluxogramas apresentados nas figuras 24 a 32. Nesses fluxogramas estão descritos as possíveis seqüências de procedimentos realizadas durante a execução do simulador.

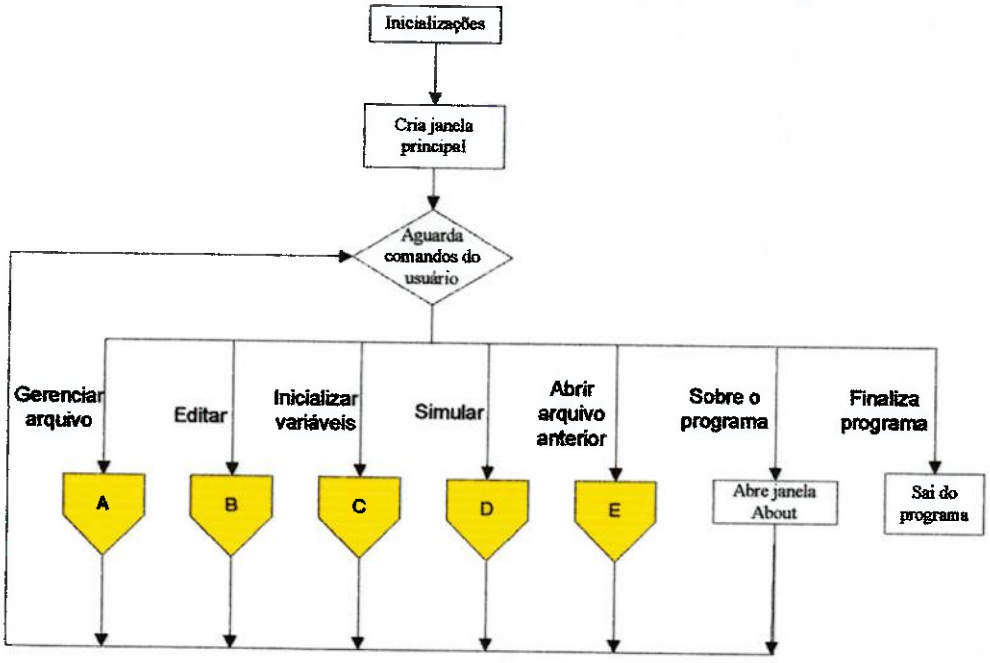

Figura 24 - Fluxograma do funcionamento interno do simulador

Ao ser inicializado, o simulador cria a janela principal de interface com o usuário e aguarda algum comando do mesmo, que podem ser as opções ilustradas na Figura 24.

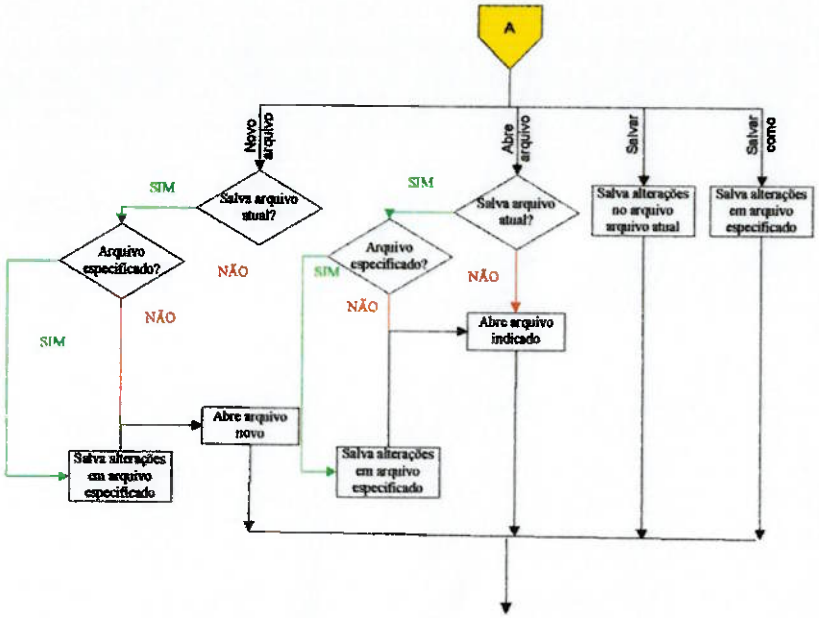

Figura 25 - Fluxograma detalhado da opção Gerenciar Arquivo

Dentro da opção de Gerenciamento de Arquivos, as opções ilustradas na Figura 25 podem ser realizadas.

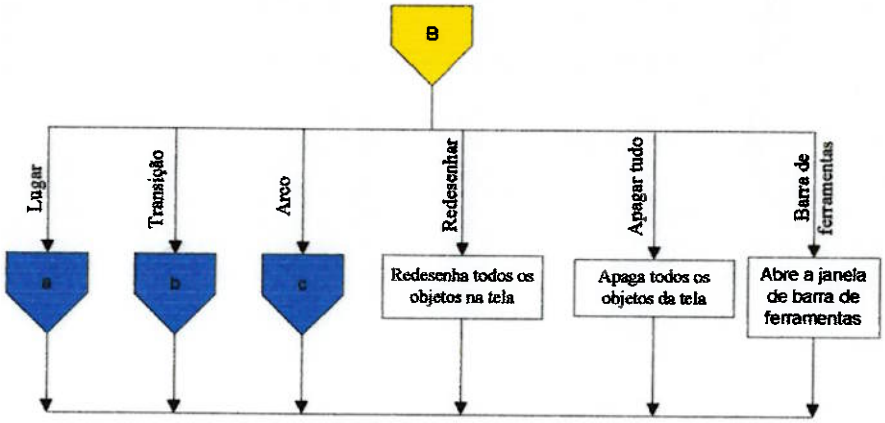

Figura 26 - Fluxograma detalhado da opção Editar

Na opção Editar, as opções ilustradas na Figura 26 podem ser utilizadas. A Figura 27 , descreve a opção Lugar, a Figura 28, a opção Transição e a Figura 29, a opção arco.

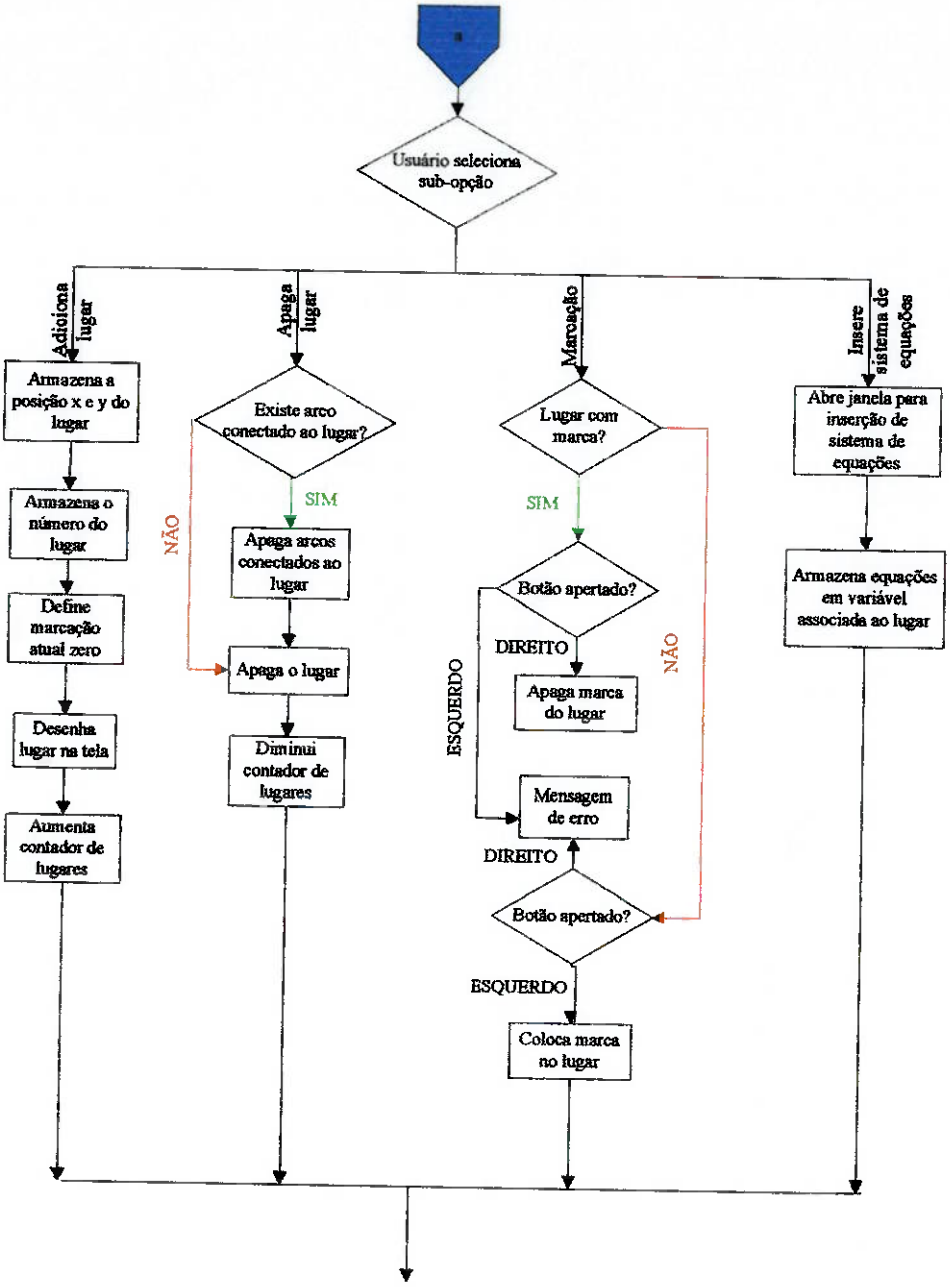

Figura 27 – Fluxograma detalhado da opção Lugar

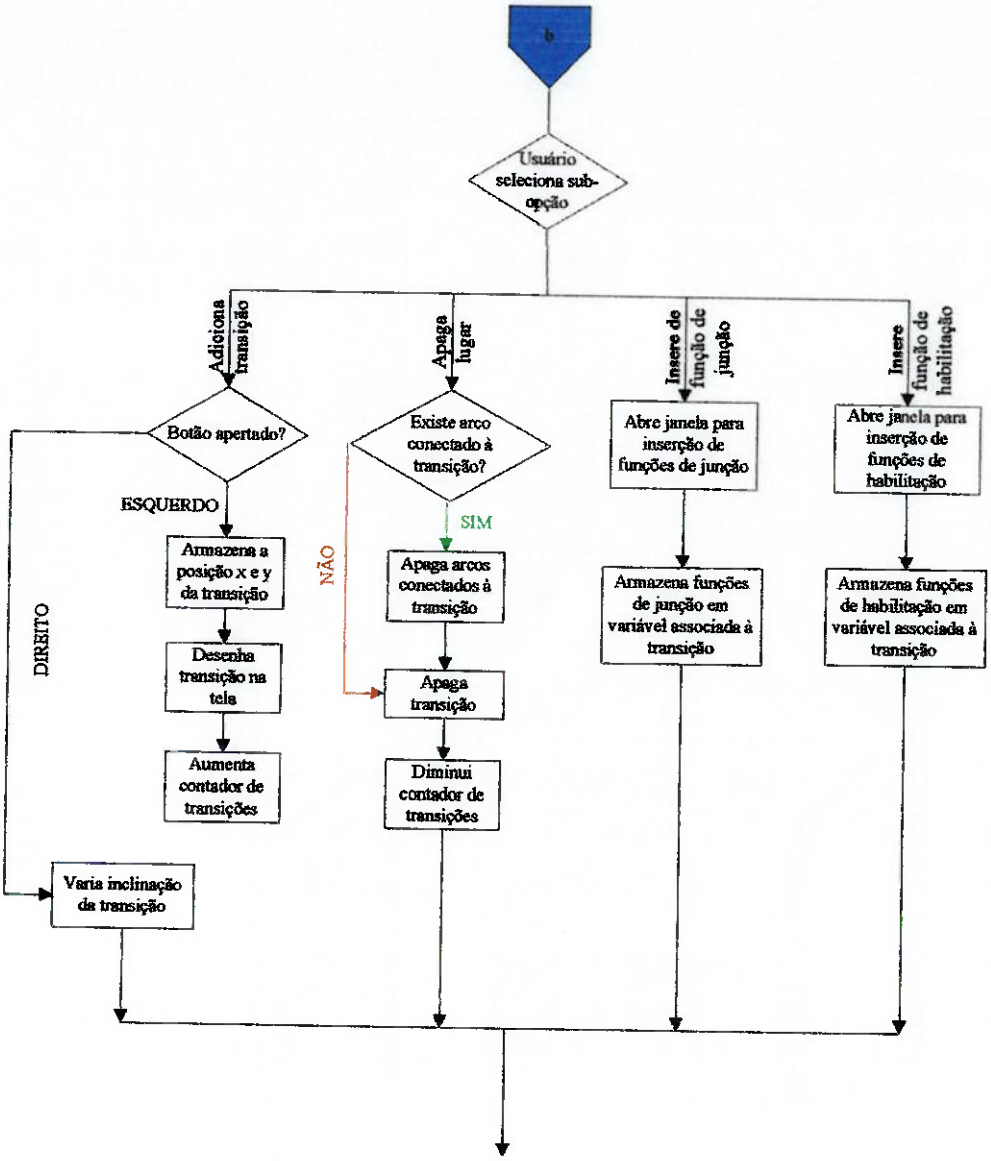

Figura 28 - Fluxograma detalhado da opção Transition

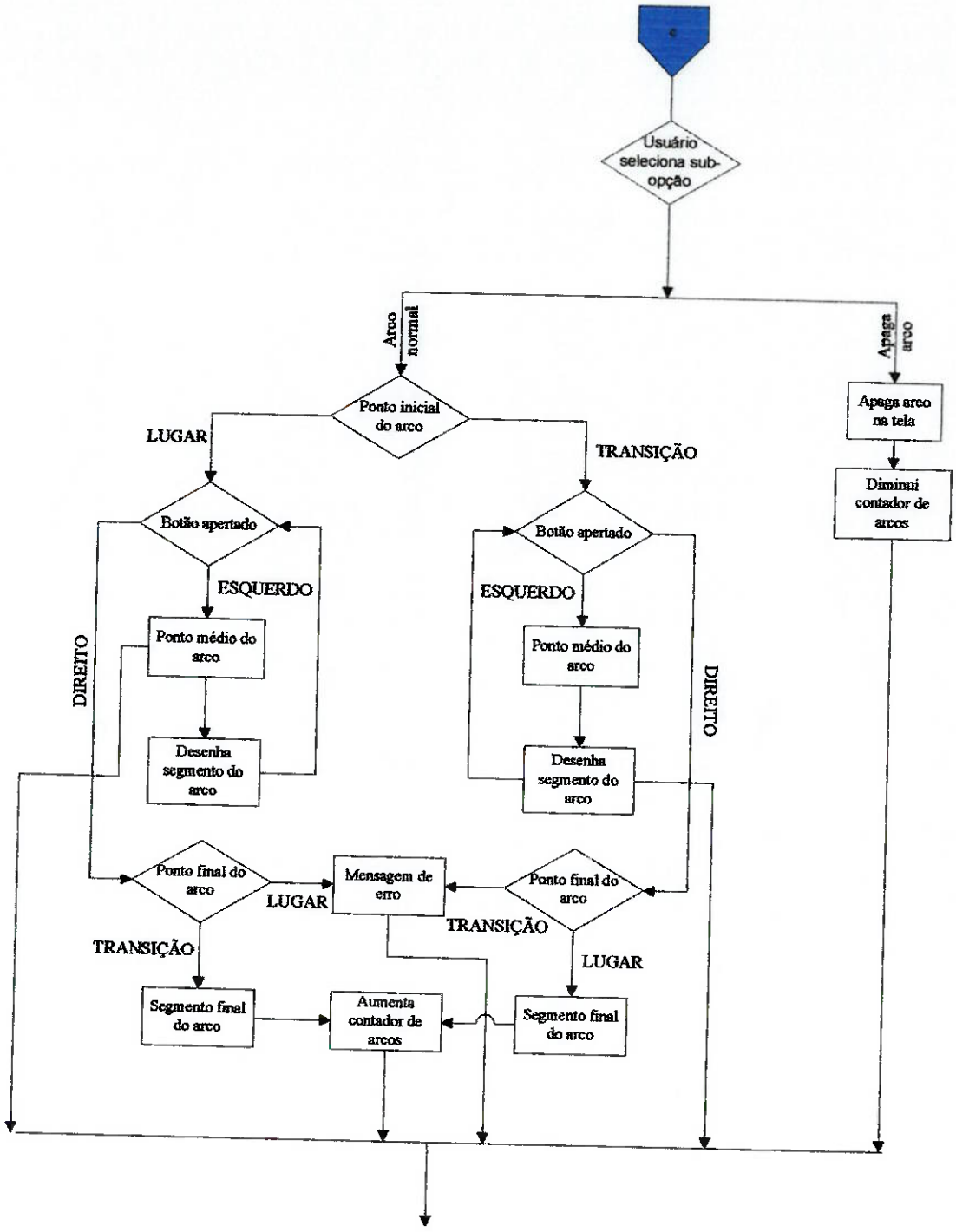

Figura 29 - Fluxograma detalhado da opção Arc

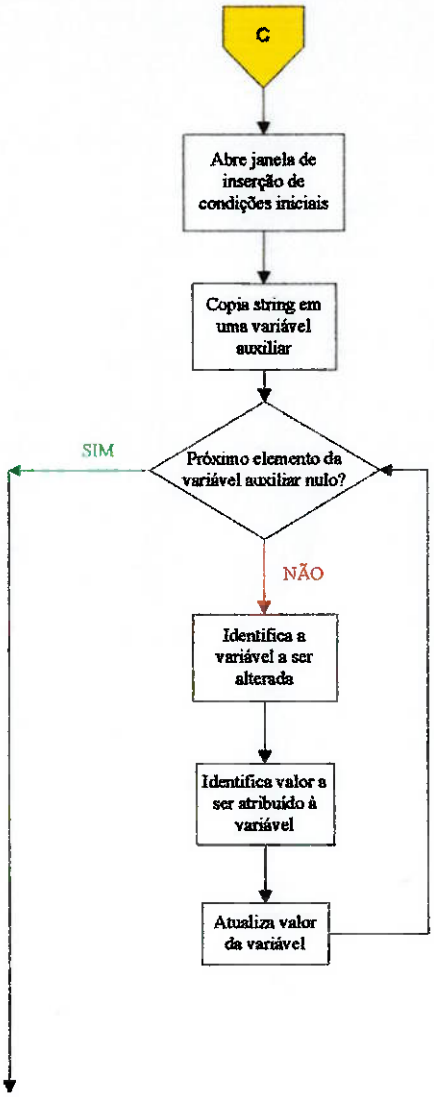

Figura 30 - Fluxograma detalhado do funcionamento do menu Variables

A Figura 30 ilustra a seqüência de ações realizadas para inserção das condições iniciais das variáveis contínuas.

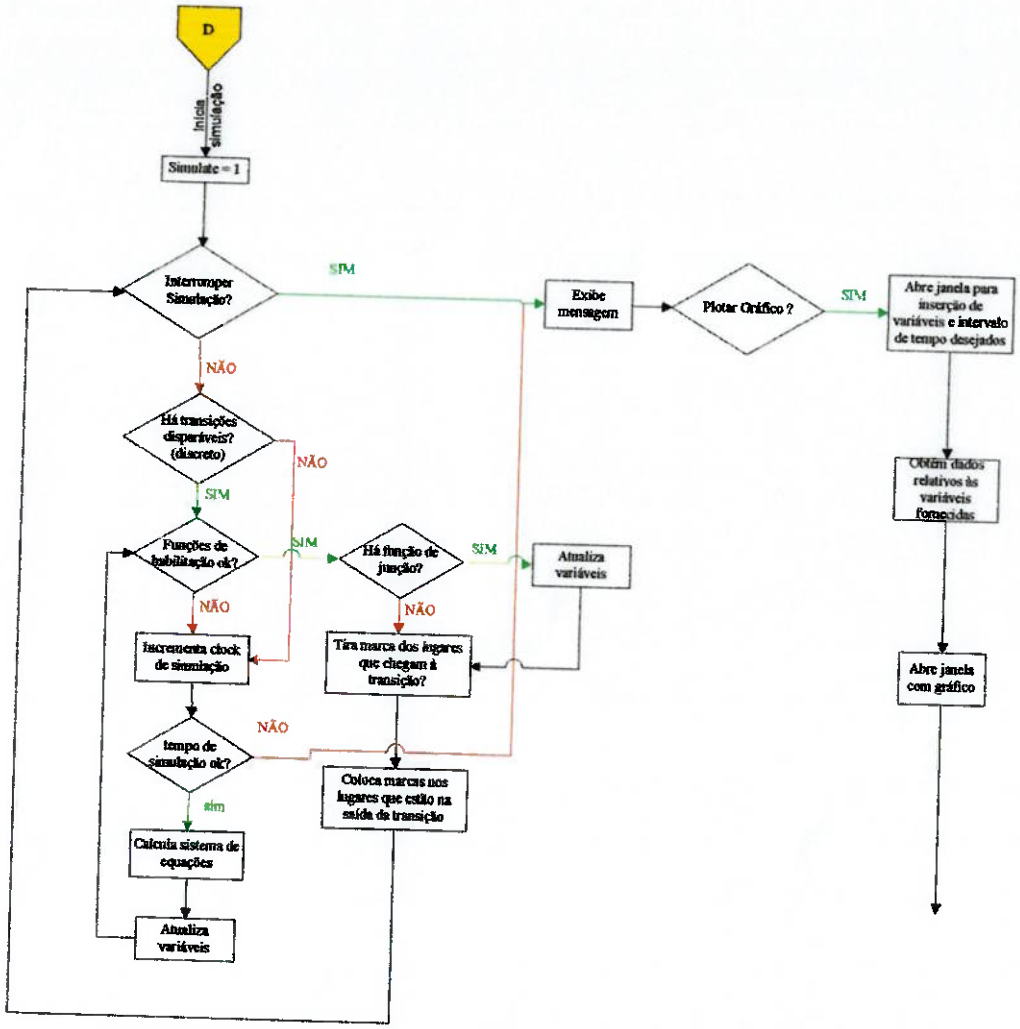

Figura 31 - Fluxograma detalhado do funcionamento do menu Simulation

A Figura 31 ilustra a seqüência de ação para realizar a simulação do modelo implementado.

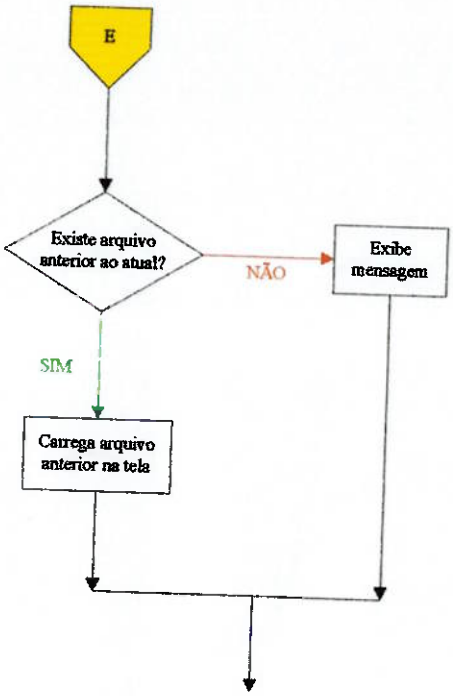

Figura 32 - Fluxograma detalhado do funcionamento do menu Previous File

A Figura 32 ilustra os procedimentos para abertura do arquivo utilizado anteriormente.

3.4 Interface com o usuário

Quando o Simulador de Sistema Híbrido é iniciado, uma tela inicial á aberta com os principais menus de navegação como mostrado na Figura 33.

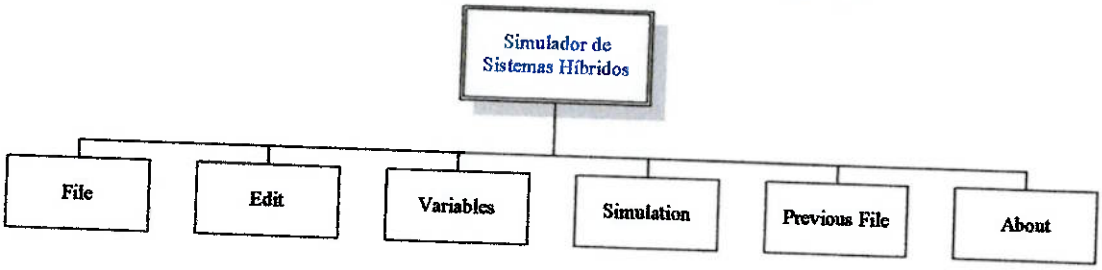

Figura 33 - Menus do simulador

Na figura 34 é ilustrado a tela inicialmente apresentada ao usuário do simulador.

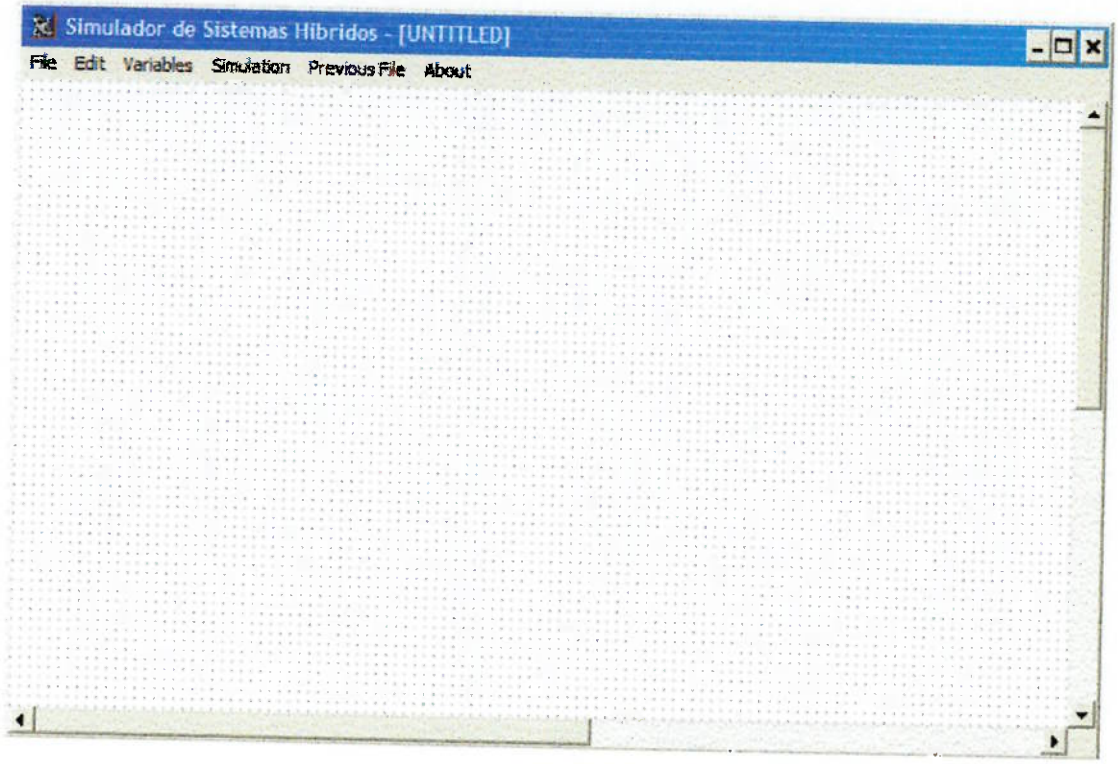

Figura 34- Tela de início do simulador

Na tela de início no simulador, o menu disponibiliza ao usuário as seguintes opções:

- File – opções de arquivo;
- Edit – opções de edição;
- Variables – inicialização das variáveis contínuas;
- Simulation – opções de simulação;
- Previous File – retornar ao arquivo anterior;
- About – informações sobre o simulador.

3.4.1 Sub-menu File

No Sub-menu File, o usuário pode realizar as seguintes ações:

- iniciar um novo arquivo (New);
- abrir um arquivo já existente (Open);
- salvar todos os detalhes do documento no arquivo atual (Save);
- salvar todos os detalhes do documento em outro arquivo (Save As);
- sair do programa (Exit);

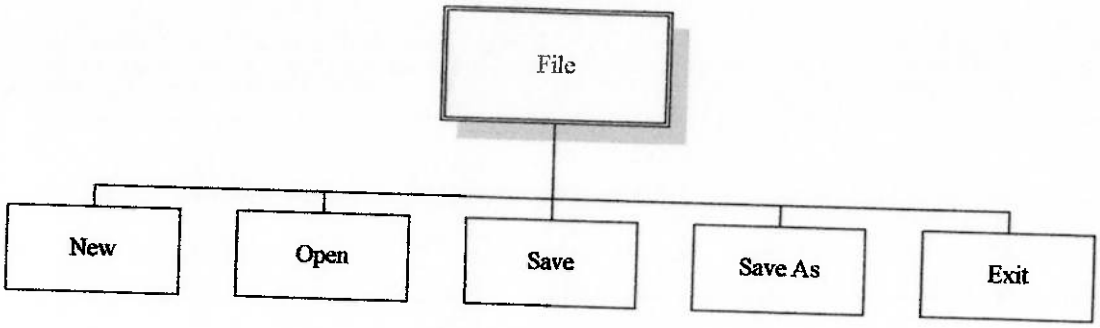

Figura 35 - Sub-menus de File

3.4.2 Sub-menu Edit

No sub-menu Edit, o usuário pode realizar as seguintes ações, mostradas na Figura 39:

- Place:
  - Add: para adicionar os lugares na tela;
  - Delete: para apagar os lugares;
  - Token: para inserir as marcas nos lugares;
  - Equations: para adicionar o sistema de equação associado ao lugar. O usuário deve entrar as equações finalizadas por ponto e vírgula, sendo que para equações diferenciais, a variável alterada deve ser identificada como *dxxx*. No caso de equações algébricas, como *uxxx*;
- Transition:
  - Add: para adicionar a transição na tela.
  - Delete: para apagar a transição;
  - Juction Function: para inserir uma função de junção na transição. As funções de junção devem ser inseridas pelo usuário separadas e finalizadas por ponto e vírgula;
  - Enable Funciton: para inserir a função de habilitação na transição. As funções de habilitação também devem ser separadas e finalizadas por ponto e vírgula quando inseridas pelo usuário;
- Arc:
  - Normal: para inserir um arco normal na tela;
  - Delete: para apagar o arco;
- Redraw: para redesenhar a tela.

- Delete All: para apagar todos os elementos da tela.
- Edit Tool Bar: para exibir a barra de ferramenta para construção e edição da rede híbrida, a barra para a opção Place é ilustrada na Figura 36, para Transition na Figura 37 e para Arc na Figura 38.

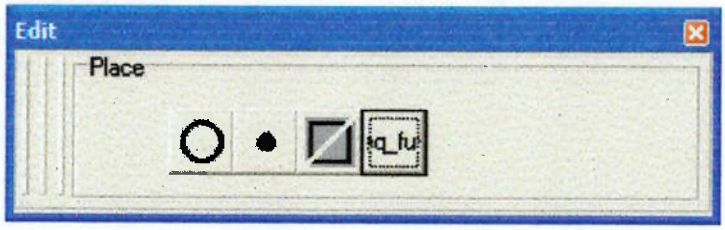

Figura 36 - Barra de ferramentas - Place

Figura 37 - Barra de ferramentas – Transition

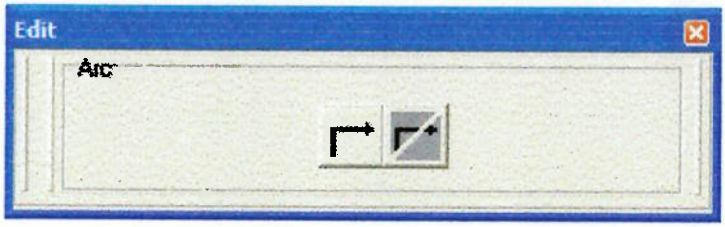

Figura 38 - Barra de ferramentas - Arc

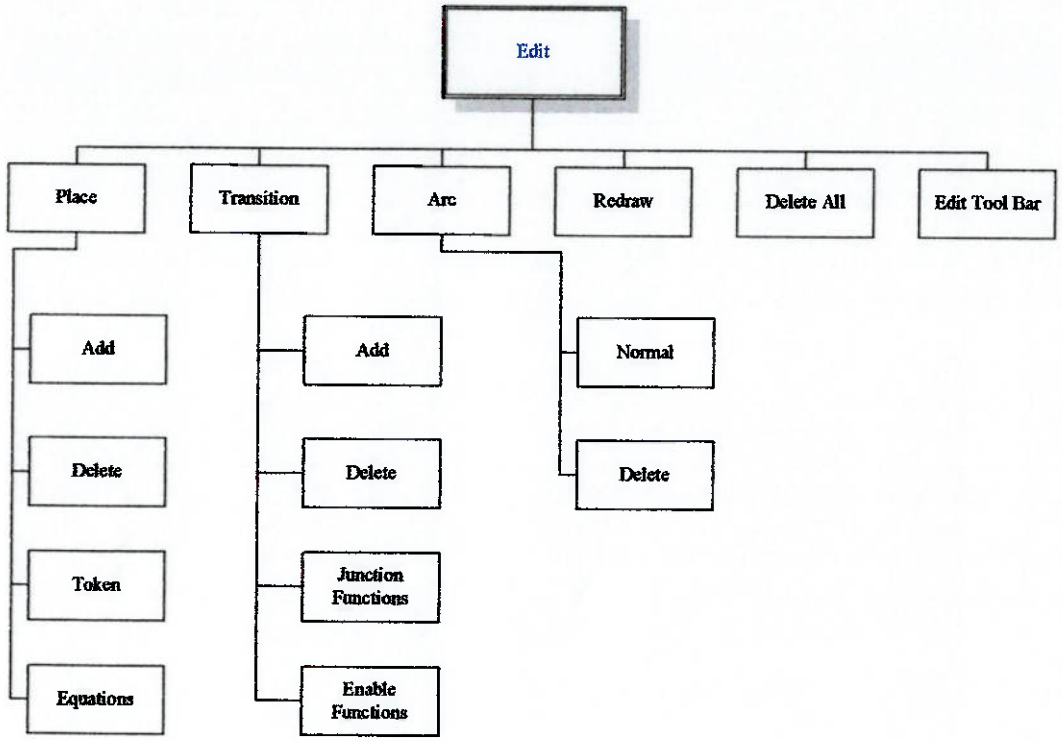

Figura 39 - Sub-menu Edit

3.4.3 Sub-menu Variables

Nesta opção o usuário pode inserir as condições iniciais das variáveis contínuas através de uma janela apresentada na da Figura 40 (estrutura das funções na figura 41). As variáveis devem ser identificadas por *uxxx*, variando de 000 a 099 (ex: u001, u002, etc.).

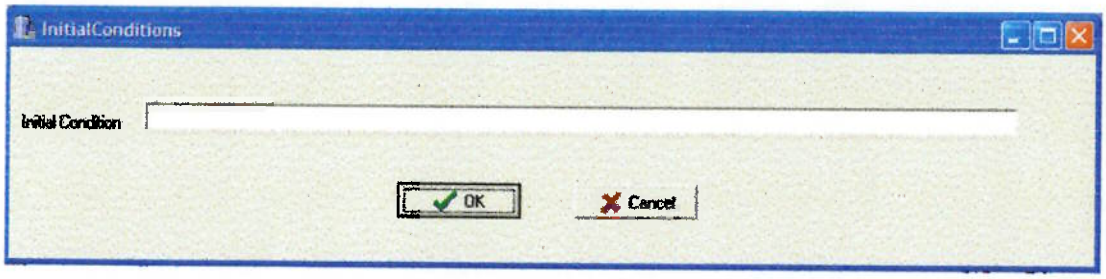

Figura 40 - Janela para inserção das condições inicias

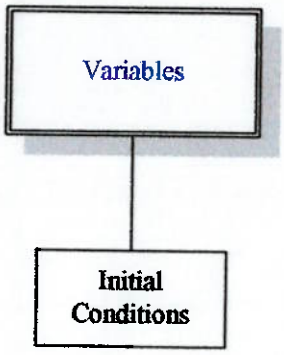

Figura 41 - Sub-menu Variables

3.4.4 Sub-menu Simulation

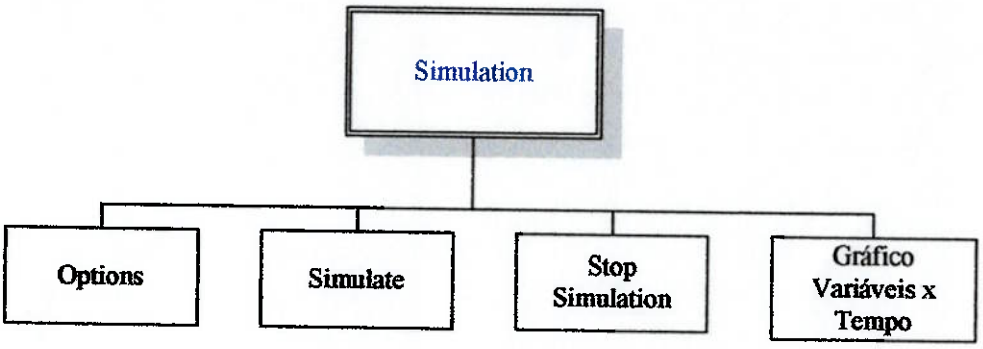

Figura 42 - Sub-menu Simulation

No sub-menu da Figura 42 , o usuário pode realizar as seguintes ações:

- Options: para configurar as seguintes opções de simulação através da janela exibida na Figura 43.
    - Delay Time: para definir o intervalo para visualização do estado da rede entre o disparo de transições durante a simulação:
        - None;
        - Short;
        - Normal;
        - Long;
        - Push button;
    - Fire Transition: para definir o critério para disparo quando mais de uma transição está habilitada.
        - Sequential: para disparar uma transição por vez, na seqüência em que foram criadas.

- Random: para disparar uma transição por vez, em seqüência aleatória.
- Push button: para disparar apenas a transição que for selecionada via mouse.
- Regular: para disparar todas as transições habilitadas em um dado instante, simultaneamente.

o Conflict: para determinar o modo de resolução de conflitos.

- Sequential: para escolher uma das transições conflitantes, a prioridade é dada pela sua posição na lista ligada.
- Random: para escolher uma das transições conflitantes, sem prioridade.
- Push button: para disparar a transição que for selecionada via mouse.
- Normal: para verificar as funções de habilitação das transições em conflito, disparando primeiro a que satisfizer a função.

A figura 43 ilustra a tela para definição das opções de simulação.

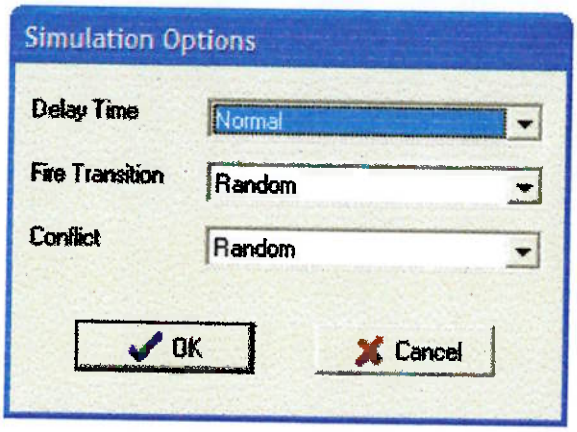

Figura 43 - Janela de opções de simulação

- Simulation: para iniciar ou continua a simulação da rede.
- Stop Simulation: para interromper a simulação.
- Gráfico Variáveis x Tempo: para desenhar o gráfico de variáveis escolhidas pelo usuário, do início da simulação até um tempo final também determinado pelo usuário, que pode optar pela escala em segundos (s), minutos (m) ou horas(h),

indicando a escolha no segundo box. As variáveis também devem ser separadas e finalizadas por ponto e vírgula. A janela para inserção dessas opções está ilustrada na Figura 44.

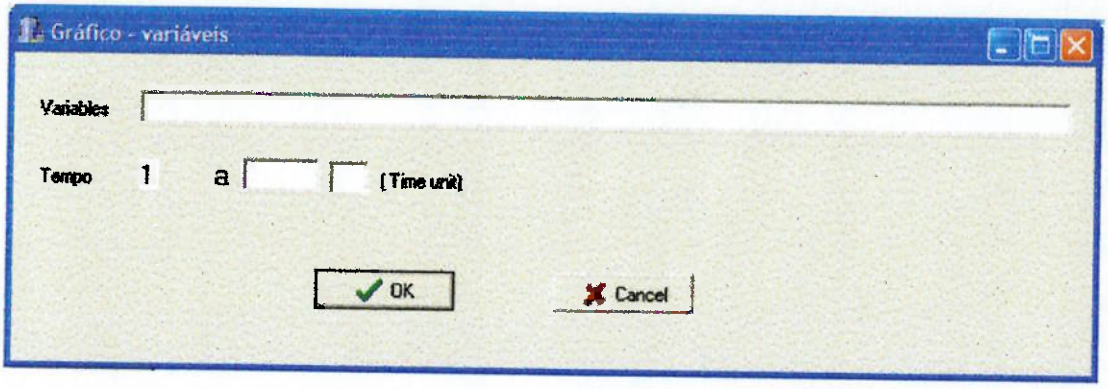

Figura 44 - Janela para inserção de opções do gráfico

A Figura 45 ilustra a tela de saída resultante do acionamento da função GraficoVariaveis x Tempo.

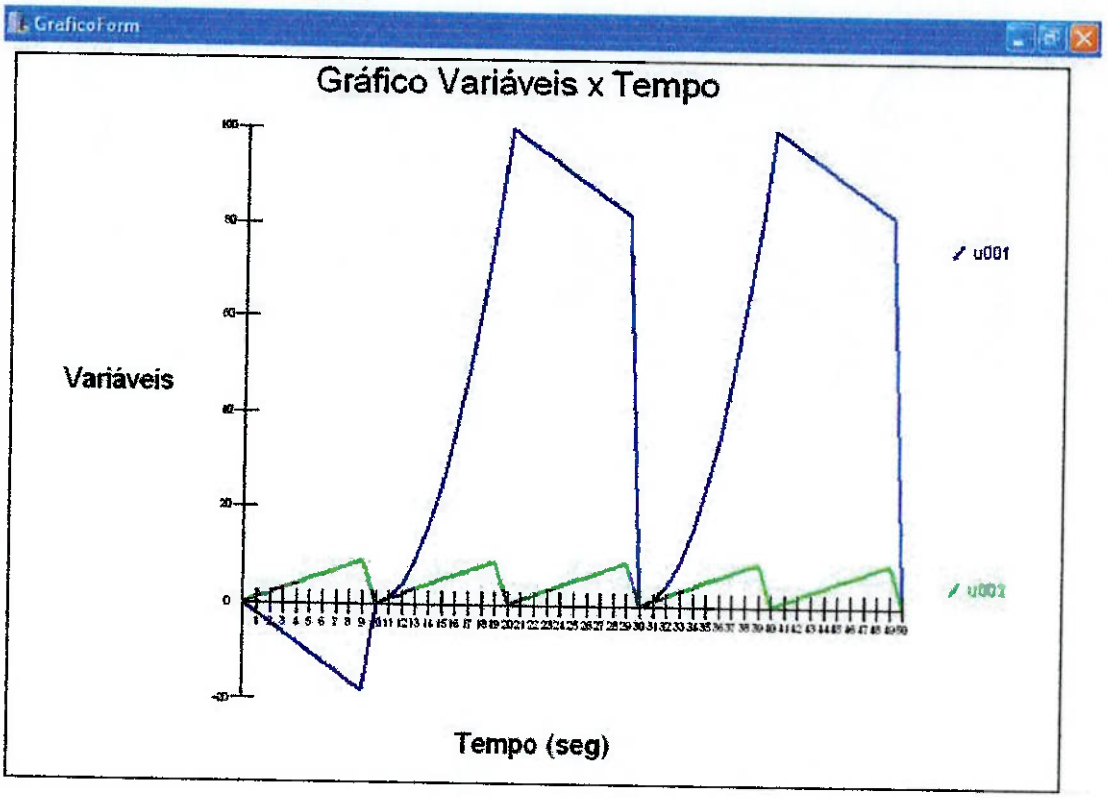

Figura 45 - Exemplo de gráfico

3.4.5 Sub-menu Previous File

Este comando do menu principal permite abrir qualquer um dos arquivos editados anteriormente na seqüência inversa de utilização.

3.4.6 Sub-menu About

Este comando do menu principal abre uma janela, ilustrada na Figura 46, com informações sobre o *software* desenvolvido.

Figura 46 – Janela de informações do *software*.

4 EXEMPLO DE APLICAÇÃO

Neste capítulo é apresentado um exemplo de aplicação em que um sistema é modelado e simulado no Simulador de Sistemas Híbridos implementado.

4.1 Sistema de ar condicionado

O exemplo de aplicação corresponde ao Sistema de Ar Condicionado ilustrado na Figura 47. Este sistema tem as seguintes características:

- Todo ar insuflado no ambiente é proveniente do exterior, não ocorre renovação de ar do ambiente.
- O fluxo de ar é imposto por um ventilador que pode assumir duas velocidades distintas. Sendo que a maior delas é utilizada no caso de detecção de fumaça no ambiente.
- O ar a ser insuflado é resfriado por uma serpentina de água fria.
- Um controlador do tipo On/Off abre e fecha a válvula de água fria da serpentina de acordo com a temperatura no ambiente.
- Considera-se duas fontes de carga térmica no ambiente: pessoas e equipamentos.

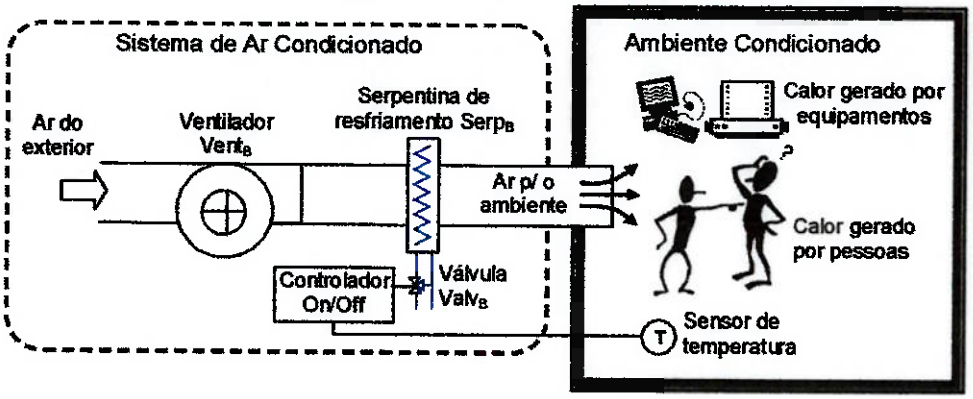

Figura 47 - Sistema de Ar Condicionado.

O Sistema de Ar Condicionado possui um Sistema Supervisório que tem as seguintes funções:

- Atuar sobre os equipamentos de acordo com o modo de operação selecionado pelo usuário;
- Permitir ao técnico de ar condicionado selecionar a temperatura desejada no ambiente;

- Supervisionar o funcionamento da serpentina, identificando eventuais vazamentos e notificando o técnico do ar condicionado;
- Em caso de incêndio (informado pelo sistema de gerenciamento do edifício), modificar a velocidade do ventilador e, no caso do sistema A, posicionar a caixa de mistura para 100% de renovação.

4.2 Modelo do Sistema

O modelo deste sistema é apresentado a seguir. Para facilitar a compreensão a apresentação modelo está dividida em 'módulos'. A composição dos módulos é feita através da fusão das transições com nomes semelhantes.

- Módulo 1 (Figura 48): representa o funcionamento do ventilador.

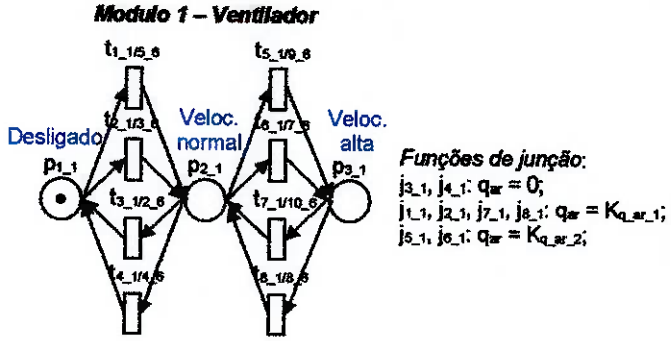

Figura 48 – Módulo 1 – Ventilador.

- Módulo 2 (Figura 49): representa o funcionamento da serpentina. A modelagem da troca de calor quando a válvula da serpentina está aberta é feita de acordo com o equacionamento apresentado em [Villani, 2000].

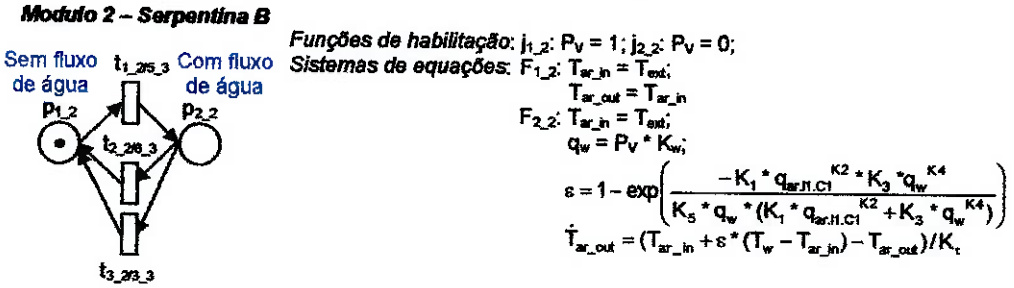

Figura 49 - Módulo 2 – Serpentina.

- Módulo 3 (Figura 50): representa a operação do controlador ON/OFF da serpentina.

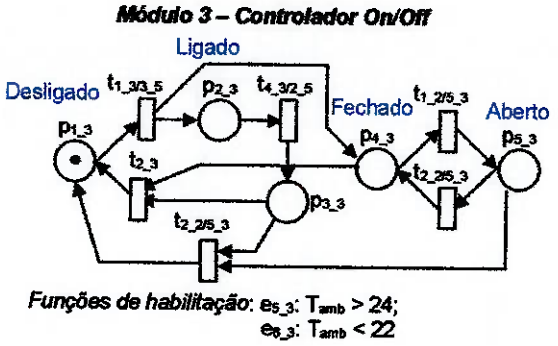

Figura 50 – Módulo 3 - Controle On/Off.

- Módulo 4 (Figura 51): representa a evolução da temperatura no ambiente condicionado em função do número de pessoas (N_p), do número de equipamentos ligados (N_{eq}) e da carga térmica retirada pelo ar condicionado (Q_{ac}).

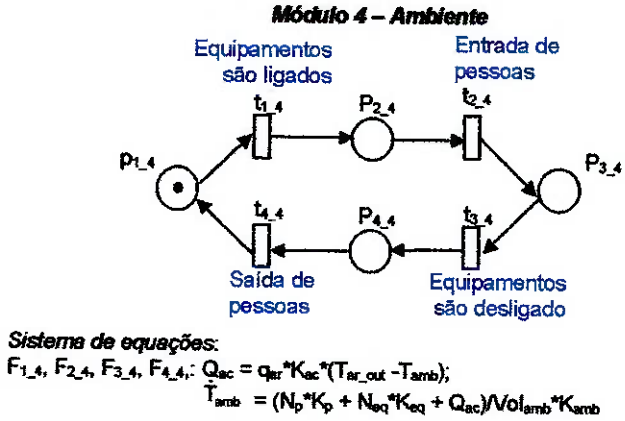

Figura 51 – Módulo 4 - Ambiente

- Módulo 5 (Figura 52): representa a parcela do interface entre o Sistema Supervisório responsável por interagir com o Controlador On/Off.

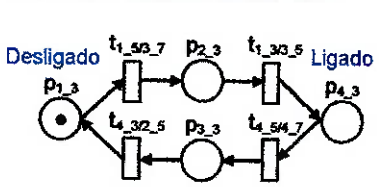

Figura 52 - Rede: Módulo 5 - Controlador

- Módulo 6 (Figura 48): representa a parcela do interface entre o Sistema Supervisório responsável por interagir com o Ventilador.

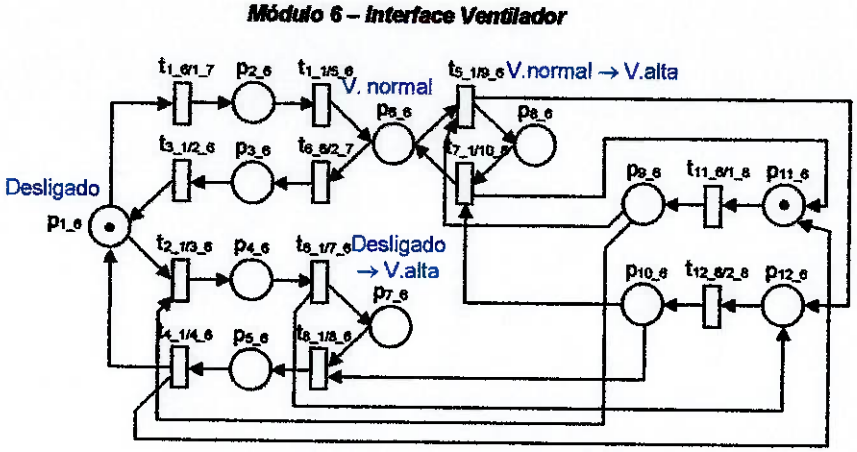

Figura 53 - Módulo 6 - Interface Ventilador

- Módulo 7 (Figura 48): representa a parcela do interface entre o Sistema Supervisório que permite aos usuários do ambiente considerado escolher entre 3 modos de operação: ar condicionado desligado, apenas ventilação e com resfriamento.

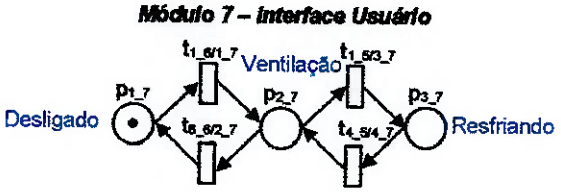

Figura 54 - Módulo 7 - Interface Usuário

- Módulo 8 (Figura 48): representa a parcela do Sistema Supervisório que interage com o sistema de detecção de fumaça do Sistema de Gerenciamento do Edifício.

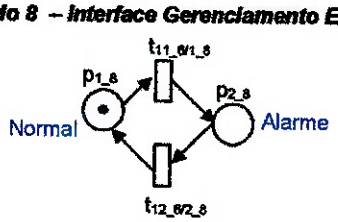

Figura 55 - Módulo 8 - Interface Gerenciamento Edifício

O estado inicial do sistema é dado por:

- Marcação inicial:

Módulo 1: O ventilador está no estado desligado marca no lugar P1_1.

Módulo 2: A serpentina está sem fluxo de água marca no lugar P1_2.

Módulo 3: O controle ON/OFF está desligado marca no lugar P1_3.

Módulo 4: No módulo ambiente a marca está no lugar P1_4, que representa o ambiente sem nenhum equipamento ligado e sem nenhuma pessoa.

Módulo 5: A marca no lugar P1_3 representa a situação de controlador desligado.

Módulo 6: Neste módulo a marcação inicial está nos lugares P1_6, e P11_6 que representa a situação de ventilador desligado.

Módulo 7: A marca inicial está no lugar P1_7 que representa que a interface com o usuário desligada.

Módulo 8: A marca inicial está no lugar P1_8 representando uma situação configuração normal, sem incêndio.

- Valor inicial das variáveis contínuas são:

$K_{q\_ar\_1} = 0{,}7m^3/s$

$K_{q\_ar\_2} = 1{,}4m^3/s$

$q_{ar} = 0$

$q_w = 0$

$P_v = 0$

$T_{ext} = 27°C$

$T_w = 7°C$

$\varepsilon = 0$

$K_1 = 6464$

$K_2 = 0{,}8$

$K_3 = 10^7$

$K_4 = 4{,}15 \cdot 10^6$

$K_w = 6{,}8 \cdot 10^{-4} m^3/s$

$K_\tau = 300s$

$K_{ac} = 1{,}2987 \cdot 10^3$

$N_p = 0$

$K_P = 100W$

$N_{eq} = 0$

$K_{eq} = 20W$

$Vol_{amb} = 500m^3$

$K_{amb} = 1{,}2987 \cdot 10^3$

4.3 Estabelecimento de cenários

O modelo apresentado não especifica o comportamento do usuário, do ambiente exterior (de onde o ar a ser insuflado provém) ou do sistema de gerenciamento do edifício, mas apenas mostra como o Sistema de Ar Condicionado reage à interferência destes elementos.

Para que a simulação possa ser realizada é necessário que a relação destes elementos com o Sistema de Ar Condicionado seja definida, isto é, que seja estabelecido um 'cenário'(experimento), onde especifica-se:

- Quando a ventilação é ligada/desligada.
- Quando o resfriamento é ligada/desligada.
- Quantos equipamentos são ligados/desligados e quando.
- Quantas pessoas entram/saem do ambiente e quando.
- Qual o comportamento da temperatura exterior (variável T_{ext})
- Se ocorre ou não detecção de fumaça e quanto tempo o alarme de incêndio permanece ligado.

Esta especificação é feita através da adição de funções de habilitação, junção e de sistemas de equações ao Módulo 2 – Serpentina, ao Módulo 7 – Interface Usuário e ao Módulo 8 – Interface Gerenciamento do Edifício. Um exemplo é apresentado a seguir.

4.4 Simulação

4.4.1 Definição do Cenário (Experimento)

Como exemplo para simulação, considerou-se o seguinte cenário:

- Temperatura exterior constante em 27°C.
- Disparo do alarme de incêndio em t = 30 min:
  - Adição da função de habilitação $e_{11\_6/1\_8}$: t=1800 ao Módulo 6 e Módulo 8.
- Retorno das condições normais (sem incêndio,fumaça) em t = 60 min:
  - Adição da função de habilitação $e_{12\_6/2\_8}$: t=3600 ao Módulo 6 e Módulo 8.
- Ativação de 10 equipamentos no instante t=120min:
  - Adição da função de junção e_{1_4}: Ne=10 ao Módulo 4.
  - Adição da função de habilitação e_{1_4}: t=7200 ao Módulo 4.

- Ativação ventilação no instante t=180min:
  - Adição da função de habilitaçao $e_{1\_1/5\_6}$: t=10800 ao Módulo 1 e Módulo 6.
- Entrada de uma pessoa no ambiente no instante t=240min:
  - Adição da função de junção e_{2_4}: Np=1 ao Módulo 4.
  - Adição da função de habilitação e_{2_4}: t=14400 ao Módulo 4.
- No instante t=300min é ligado o resfriamento:
  - Adição da função de habilitação $e_{1\_5/3\_7}$: t=18000 ao Módulo 5 e Módulo 7.
- No instante t=480min é desligado o resfriamento:
  - Adição da função de habilitação $e_{4\_5/4\_7}$: t=28800 ao Módulo e Módulo 7.
- Os 10 equipamentos são desligado no instante t=540min:
  - Adição da função de junção e_{3_4}: Ne=0 ao Módulo 4.
  - Adição da função de habilitação e_{3_4}: t=32400 ao Módulo 4.
- No instante t = 600 min a ventilação é desligada:
  - Adição da função de habilitaçao $e_{6\_6/2\_7}$: t=36000 ao Módulo 1 e Módulo 6.
- No instante t=660min uma pessoa sai do ambiente:
  - Adição da função de junção e_{4_4}: Np=0 ao Módulo 4.
  - Adição da função de habilitação e_{4_4}: t=39600 ao Módulo 4.

4.4.2 Execução da simulação

Do ponto de vista discreto, a simulação comprova na seguinte evolução (ocorrência de eventos):

- De t=0 a t=1800: não ocorre nenhum evento, a marcação permanece igual a marcação inicial: ventilador desligado, controle on/off desligado, alarme de incêndio desligado e serpentina sem fluxo de água;
- Em t=1800: ocorre o disparo do alarme de incêndio, o ventilador é ligado em velocidade alta;
- Em t=3600: ocorre o retorno às condições normais, o alarme de incêndio é desligado e o ventilador também.
- Em t=7200: 10 equipamentos são ligados no ambiente
- Em t=10800: O modo ventilação é ativado: liga-se o ventilador na velocidade normal.

- Em t=14400: ocorre a entrada de 1 pessoa no ambiente.
- Em t=18000: O modo de operação 'resfriamento' é ativado: controlador ligado, controle on/off ligado, serpentina com fluxo de água.
- Entre o tempo t=18000 a t=28800: ocorre a alternância entre a situação da serpentina com e sem fluxo de água até se desligar o resfriamento.
- Em t=28800: O modo de operação 'ventilação' é ativado: resfriamento é desligado, controlador é desligado, controle on/off é desligado e serpentina fica sem fluxo de água.
- Em t=32400: Os 10 equipamentos são desligados.
- Em t=36000: O modo de operação 'desligado' é ativado: desliga-se o ventilador.
- Em t= 39600: Saída de 1 pessoa do ambiente.
- Em t= 43200: Fim da simulação.

Pode-se visualizar a evolução das variáveis contínuas do sistema em gráficos como os ilustrados nas Figura 56 e Figura 57.

A Figura 56 mostra a evolução da temperatura no ambiente durante todo o tempo de simulação ilustrando o momento da ocorrência dos eventos citados acima, possibilitando a visualização da influência desse evento nessa variável.

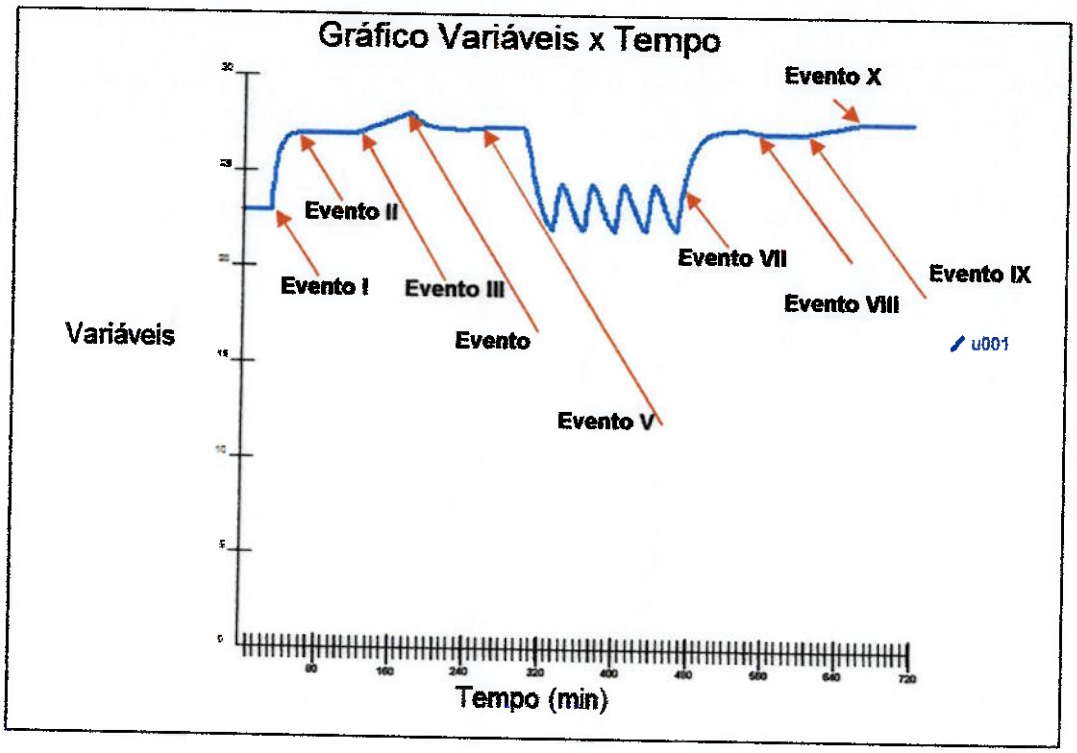

Figura 56 - Gráfico da temperatura no ambiente

A Figura 57 ilustra a evolução da temperatura no ambiente (u001) comparada com a temperatura na saída da serpentina (u002).

Pode-se perceber que a temperatura na serpentina varia somente quando o resfriamento é acionado, voltando à temperatura externa quando o resfriamento é desligado.

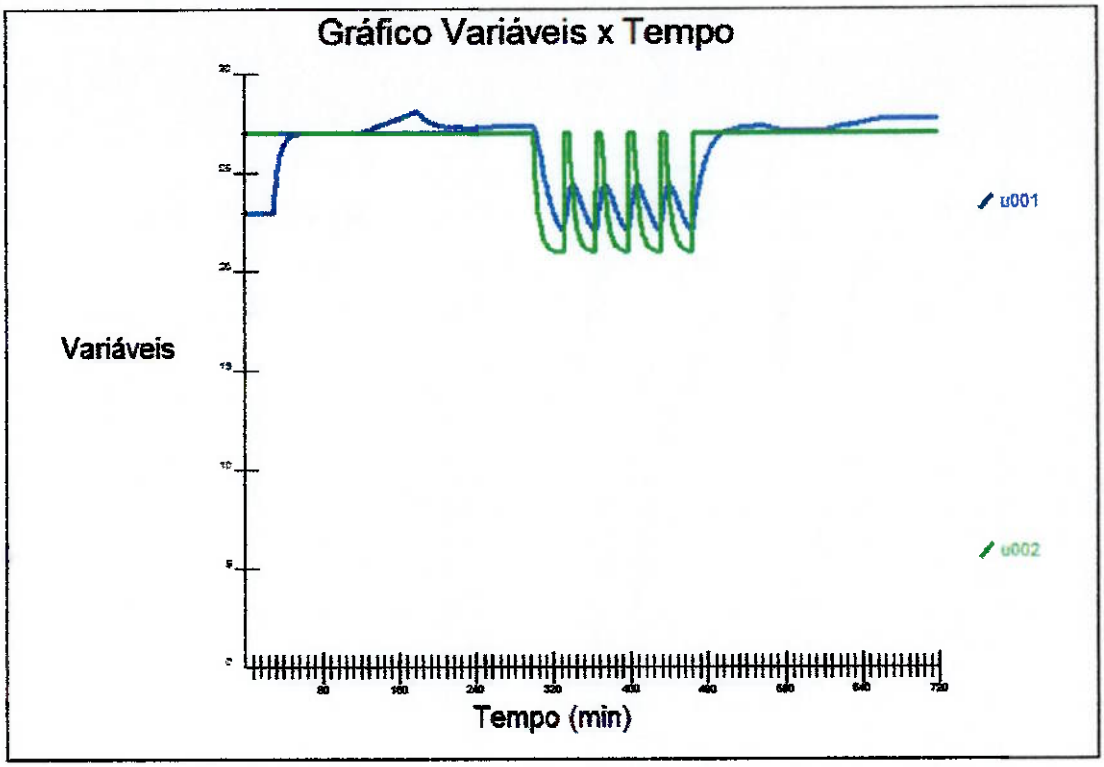

Figura 57 - Gráfico da temperatura no ambiente e na saída da serpentina

Através deste exemplo verifica que o *sofware* consegue representar muito bem um cenário onde estão contidas variáveis contínuas e discretas. Os resultados da simulação serve como base para uma prévia ou confirmação do processo que se deseja estudar. Identificando assim os pontos falhos do projeto.

5 CONCLUSÃO

Este texto apresentou os principais aspectos do desenvolvimento de um simulador de sistemas híbridos baseado em redes de Petri Predicado Transição Diferencial. Aspectos teóricos sobre análise e modelagem de sistemas a eventos discretos (SED), modelagem de sistemas de variáveis contínuas (SVC) e método de resolução de equação diferencial foram descritos.

Através da análise do exemplo de aplicação realizado no programa foi constatado que o simulador atingiu com êxito os objetivos iniciais deste trabalho.

O simulador, foi concebido para ter uma interface gráfica amigável, para ambiente Windows, garantindo a familiaridade do usuário com o ambiente e a facilidade de utilização do simulador.

O tempo de simulação em comparação ao Simulador desenvolvido por [Yen-Tsang, 2001] foi reduzido significativamente, principalmente porque o método implementado para resolver as equações diferenciais obtém a solução em um tempo menor se comparado ao Simulink, sem que ocorra qualquer tipo de perda de precisão nos resultados.

Na simulação da parte discreta o usuário consegue acompanhar visualmente a movimentação das marcas, o que ajuda bastante na verificação do modelo que se deseja simular. Este recurso gráfico também não estava disponível em [Yen-Tsang, 2001].

O desenvolvimento da solução teve como base um *software* de Rede de Petri (Petri Net Simulator, [Takada, 1999]) e que de um lado facilitou o desenvolviemento mas de outro gerou problemas, visto que o código do mesmo não estava bem documentado. Outra dificuldade encontrada foi com o compilador utilizado, Borland Builder C++, pois as versões recentes não possuem as mesmas bibliotecas das versões anteriores. Estes problemas foram sanados possibilitando o desenvolvimento do simulador.

É importante ressaltar a escassez de simuladores de sistema híbridos e o resultado deste trabalho surge como contribuição para aqueles que desejam simular modelos que envolvam SED e SVC.

6 REFERÊNCIAS BIBLIOGRÁFICAS

- Antsaklis, P. J.; Nerode A.; "Hybrid Control Systems: An Introductory Discussion to the Special Issue"; IEEE Transactions on Automatic Control, vol. 43, nº4, pp 457-459; 1998;
- Astrom, K. J.; Elmqvist, H.; Mattson, S. E.; "Evolution of continuous-time modeling and simulation". In 12th European Simulation Multiconference, Manchester, 1998;
- Cardoso, J.; Valette, R.; "Redes de Petri", Editora da UFSC, Florianópolis, 1997;
- Champagnat, R.; "Supervision des Systèmes Discontinus: Definition d'un Modèle Hybride et Pilotage en Temps-rèel"; Thèse de Doctorat; Université Paul Sabatier; Toulouse; 1998;
- Chapra, Steven C.; Canale Raymond P.; "Numerical Methods For Engineers"; McGraw-Hill Publishing Company; 1988;
- Drath, R.; "Hybrid Object Nets: An Object Oriented Concept fo Modelling Complex Hybrid Systems"; In ADMP'98 3rd International Conference on Automation of Mixed Processes, Reims, 1998;
- Futema, Jorge; "Curso Prático de Solda a Ponto"; Mafersa S.A.; 1997;
- Henzinger, T. A.; "HyTech: A model checker for hybrid systems"; Software Tools for Technology Transfer, vol. 1, pp 110-122; 1997;
- Klander, Lars; Jamsa, Kris; "Programando em C/C++: a Bíblia"; Makron Books, 1999;
- Maciel, P. R. M.; "Introdução as Redes de Petri e Aplicações"; Editora Unicamp, 1996;
- Matsumoto, Élia Yathie; "Matlab 6.5 – Fundamentos de Programação"; Editora Érica; São Paulo, 2002;
- Matsusaki, Cristina Toshie Motohashi; Redes F-MFG (Functional Mark Flow Graph) e sua aplicação no projeto de sistemas antropocêntricos. Dissertação de Mestrado, EPUSP, São Paulo, 1998;

- Miyagi, Paulo Eigi; "Controle Programável – Fundamentos do Controle de Sistemas a Eventos Discretos"; Editora Edgard Blücher LTDA; São Paulo, 1996;
- Rice, John R.; "Numerical Methods, Software and Analysis", Academic Press, Inc., 1993;
- Silva, B. I.; "Modeling and Verifying Hybrid dynamic systems using checkmate"; pp 323-328; In Proc. 4th International Conference on Automatic of Mixed Processes: Hybrid Dynamic Systems, 2000;
- Silva, B. I.; "An Assessment of the Current Status of Algorithmic Approaches to the Verification of Hybrid Systems"; In 40th IEEE Conference on Decision and Control, Orlando, 2001;
- Takada, Fabio Noriyuki; "Editor e simulador de sistemas a eventos discretos baseado em Redes de Petri"; Trabalho de Formatura, EPUSP, São Paulo, 1999;
- Vetterling, William T.; Teukolsky, Saul A.; Press, William H.; Flannery, Brian P.; "Numerical Recipes – Example Book (C) Second Edition", Cambridge University Press, 1988;
- Villani, Emília; "Abordagem híbrida para modelagem de sistemas de ar condicionado em edifícios inteligentes"; Dissertação de Mestrado, EPUSP, São Paulo, 2000;
- Yen-Tsang, Chen; "Simulador de Sistemas Híbridos"; Trabalho de Formatura, EPUSP, São Paulo, 2001;