UM AMBIENTE BASEADO EM CONHECIMENTO PARA O PROJETO DE SISTEMAS DE EVENTOS DISCRETOS

Carlos J.P. Lucena (*)
Depto. de Ciência da Computação
Pontifícia Universidade Católica
Rio de Janeiro

José Reinaldo Silva (**)
Depto. de Engenharia Mecânica
Universidade de São Paulo

Italo Santiago Vega
LSD/FDTE
Universidade de São Paulo

Marcelo Finger
Depto. de Engenharia de Eletricidade
Universidade de São Paulo

Julio Arakaki
Depto. de Eng. Mecânica/LAS
Universidade de São Paulo

Resumo – Este artigo apresenta um ambiente baseado em conhecimento (o AIPUSP) para a automação do projeto de sistemas discretos de produção. O ambiente integra um pacote de simulação convencional e recursos gráficos para o modelamento de sistemas discretos, com ferramentas inteligentes tais como: um analisador dos resultados da simulação; um reestruturador de modelos - que pode sugerir modificações no "design" a partir do diagnóstico do analisador; e uma ferramenta para documentação das decisões e controle de versões de projeto, que permite inclusive a reutilização de "designs".

Abstract – This paper presents a knowledge based environment (the AIPUSP) for automatizing discrete event systems modelling. The environment integrates conventional simulation tools and graphic resources for the direct modelling, with other intelligent tools such as: a analyzer for the simulation results; an automatic redesigner - which suggest modifications in the original design based on the diagnostic of the analyzer; and a refinement tree which stores the documentation of design decisions and controls the project versions. This refinement tree also allows design reusability.

1. INTRODUÇÃO

O aumento da produtividade dos processos industriais depende intrinsecamente da possibilidade de automatizar o processo de modelamento e o escalonamento de sistemas de eventos discretos. Várias alternativas vem sendo propostas pelos pesquisadores da área para abordar, independentemente, os problemas: do modelamento, da verificação de correção (em geral através de ferramentas de simulação), e, recentemente, da otimização do processo de análise dos modelos.

Um outro enfoque para esta questão seria um ambiente unificado de projeto; seja através da integração das soluções para os problemas já mencionados, seja pela proposição de uma base teórica que resulte num paradigma novo, envolvendo todas as etapas do desenvolvimento do projeto de sistemas de produção.

Este artigo apresenta a especificação e a implementação de um ambiente integrado para o projeto de sistemas de manufatura, o AIPUSP, concebido para integrar um paradigma de modelamento baseado em "Mark Flow Graphs" (Hasegawa, 1987) e (Masuda, Hasegawa, 1980) - associado a uma técnica de abstração proposta por Miyagi (Miyagi, 1988) - com as ferramentas convencionais de simulação, e com dispositivos de análise de modelos baseados em técnicas de Inteligência Artificial. O sistema introduz ainda mecanismos para a documentação e reutilização de "designs" de projetos.

Em linhas gerais o AIPUSP pode ser caracterizado como sendo um sistema interativo de Classe 3, segundo a taxonomia proposta por Brown e Chandrasekaran (Brown, Chandrasekaran, 1984) para classificar os ambientes de projeto mecânico baseados em conhecimento. Segundo estes autores, um sistema é caracterizado como sendo de Classe 3 se tanto a estratégia de "design" quanto o conhecimento específico sobre os artefatos (ou sistemas) forem conhecidos antes de se iniciar o projeto.

No caso do AIPUSP, a estratégia de design é caracterizada

(*) Parcialmente financiado pela SID Informática (Projeto Estra)
(**) Parcialmente financiado pelo CNPq e IBM Brasil

Projeto de Pesquisa parcialmente apoiado pela Fundação para o Desenvolvimento Tecnológico da Engenharia (FDTE) e pela SID Informática (Projeto ESTRA).

pela utilização da técnica de refinamentos sucessivos (Mostow, 1985) sobre uma representação do projeto baseada em Redes de Petri do tipo Condição/Evento. Outra característica marcante desta estratégia é a divisão do projeto em níveis de abstração independentes, escolhidos pelo usuário, sugerindo um desenvolvimento hierárquico.

O ambiente utiliza conhecimento específico sobre o sistema (ou artefato) cuja produção se deseja modelar. Este conhecimento é colocado em um banco de dados e regras que tem funcionamento independente dos demais módulos do sistema AIPUSP, implicando que o ambiente pode, em principio, ser utilizado na prática, para modelar vários tipos de sistemas de eventos discretos, bastando para isso substituir-se o banco de dados de conhecimento específico.

A implementação do protótipo AIPUSP apresentada neste artigo é um resultado inicial de um projeto de pesquisa cujo objetivo é buscar uma metodologia para o projeto de sistemas de produção, baseada em um formalismo que possa amparar todas as partes do desenvolvimento do sistema, e que por outro lado seja acessível ao usuário.

O AIPUSP difere de outros sistemas inteligentes, como o FADES ("Factory Design") (Fischer, 1986), dedicado a sistemas de manufatura, na abordagem do problema de modelamento e escalonamento de sistemas. Por exemplo, o FADES prioriza os processos de especificação de componentes, posicionamento das máquinas, e o escalonamento do transporte dos artefatos dentro da fábrica, enquanto que o AIPUSP prioriza a análise do fluxo de itens, a análise de eficiência, e a reestruturação de modelos de sistemas discretos em geral (incluindo as células de manufatura). Outra diferença marcante é que o FADES inclui as análises de fluxo, de eficiência e a reestruturação de modelos como atributos adicionais ao sistema, enquanto o AIPUSP inclui os vários níveis de conhecimento específico sobre células de manufatura citados por Fisher (Fisher, 1986) - a saber, o conhecimento sobre o tipo de indústria, sobre as varias firmas pertencentes a um mesmo tipo de indústria, e sobre o produto final - somente como um banco de dados adicional, associado à interface do sistema.

No AIPUSP, a análise de eficiência é feita tanto com base em regras de produção, como em algoritmos procedurais, e não exclusivamente nas primeiras como no ISIS (Smith et alii., 1986) ou nos últimos como no sistema proposto por Bruno, Elia e Laface (Bruno et alii., 1986).

O desenvolvimento do sistema AIPUSP está sendo feito por um processo de prototipação rápida. Até o momento foram produzidas duas versões. Neste artigo apresentaremos os resultados obtidos na última versão produzida, após uma breve discussão sobre "Mark Flow Graphs" e sobre as técnicas de abstração propostas por Miyagi - que inspiraram o projeto inicial do AIPUSP.

2. "MARK FLOW GRAPHS" (MFG) E O "PROGRAM FLOW SCHEMA" (PFS)

2.1 Introdução

O comportamento dinâmico dos sistemas flexíveis de manufatura apresenta propriedades de assincronismo, escalonamento, concorrência e conflito que podem ser tratadas por representações baseadas em Redes de Petri, criadas especialmente para este fim. "Mark Flow Graphs" (MFG) (Hasegawa, 1987), (Masuda, Hasegawa, 1980) e (Hasegawa, Takahashi, 1984) é uma representação criada para aumentar o grau de segurança ("safeness") no modelamento de sistemas flexíveis de manufatura, e que pode também ser aplicada genericamente a sistemas de eventos discretos. Na representação MFG, o comportamento dinâmico (fluxo de controle) do sistema fica explícito, de modo que é possível estabelecer uma correspondência entre o grafo MFG e um programa de controle e simulação, escrito em uma linguagem de específica.

A linguagem PFS/MFG proposta por Miyagi (Miyagi, 1988) tem uma correspondência direta com a representação MFG e pode ser utilizada, tanto para o controle quanto para a validação do modelo representado, através de simulação. O "Program Flow Schema" (PFS) inclui, além disso, mecanismos de abstração que permitem a especificação de subsistemas antes que a sua representação MFG seja conhecida - o que sugere um desenvolvimento hierárquico da representação MFG dos sistemas de eventos discretos.

2.2 Mark Flow Graphs (MFG)

"Mark Flow Graphs" é uma representação baseada em Redes de Petri do tipo condição / evento. Os elementos básicos do MFG são:"box", transição, arcos orientados, marcas, arcos de controle e os sinais externos de saída. A figura 1 abaixo mostra a representação gráfica de cada um destes elementos.

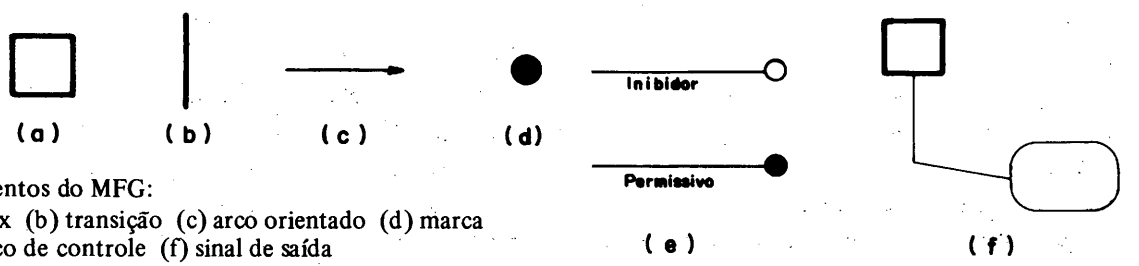


Fig.1. Elementos do MFG:
(a) box (b) transição (c) arco orientado (d) marca
(e) arco de controle (f) sinal de saída

O "box" representa uma condição, associada a um elemento do sistema e uma transição representa um evento. Os arcos orientados fazem a conexão entre um box e uma transição, e sua orientação determina o fluxo de itens e controle no sistema. Uma marca é colocada em um "box" para significar que este elemento é uma pré-condição ativa para as transições que o sucedem.

O fluxo do sistema pode ser controlado por arcos especiais (os arcos de controle), que podem permitir ou inibir uma transição. Um arco permissivo habilita uma transição conectada a ele se sua pré-condição for verdadeira (sinal ativado). Conversamente, um arco inibidor habilita uma transição se sua pré-condição for falsa (sinal desativado).

Existem dois tipos de arcos de controle: interno e externo. O arco é dito interno se o sinal de controle que ele representa provém de um S-elemento do sistema (um "box"). Quando o sinal provém de uma máquina ou subsistema não incluída no modelo MFG o arco é dito externo. Um número arbitrário de arcos podem ser conectados a um box ou a uma transição. No entanto, uma conexão múltipla entre um "box" e uma transição é proibida pela representação MFG.

O sinal externo de saída transmite o sinal do "box" para uma máquina externa, sendo 1 (ativado) quando existe uma marca no "box" e 0 (desativado), em caso contrário.

2.2.1 Condição de disparo

Uma transição se encontra habilitada (pronta para o disparo), se e somente se, ela satisfaz a todas as condições abaixo:

- não existem "boxes" de saída com marcas;
- não existem "boxes" de entrada sem marcas;
- não existem arcos permissivos com sinal 0;
- não existem arcos inibidores com sinal 1;

O disparo de uma transição provoca a remoção de todas as marcas dos "boxes" de entrada, atribuindo uma marca para cada "box" de saída. De acordo com estas regras, o número de marcas em cada "box" nunca excede 1 (garantia de segurança). A figura 1 abaixo mostra a habilitação e o disparo de uma transição.

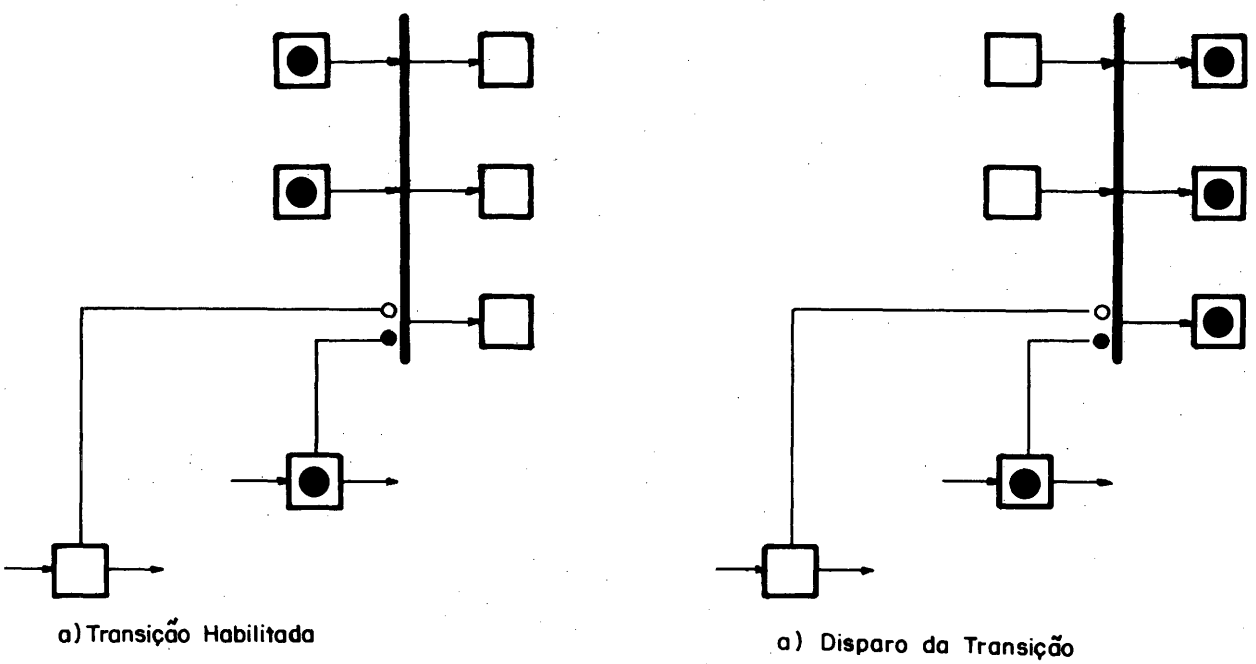


Fig.2. Representação esquemática de um disparo de Transição

2.2.2 Elementos Temporais

Uma *transição temporal* é uma transição cujo disparo é retardado de um certo intervalo de tempo depois de sua habilitação. Se a condição de disparo se extingue antes deste intervalo de tempo, a transição é desabilitada e o contador de tempo é inicializado.

O "box" temporizado representa uma condição que persiste por apenas um lapso de tempo. Uma marca em um box temporizado não pode contribuir para o disparo das transições de saída antes de decorrido um certo intervalo de tempo especificado pelo box.

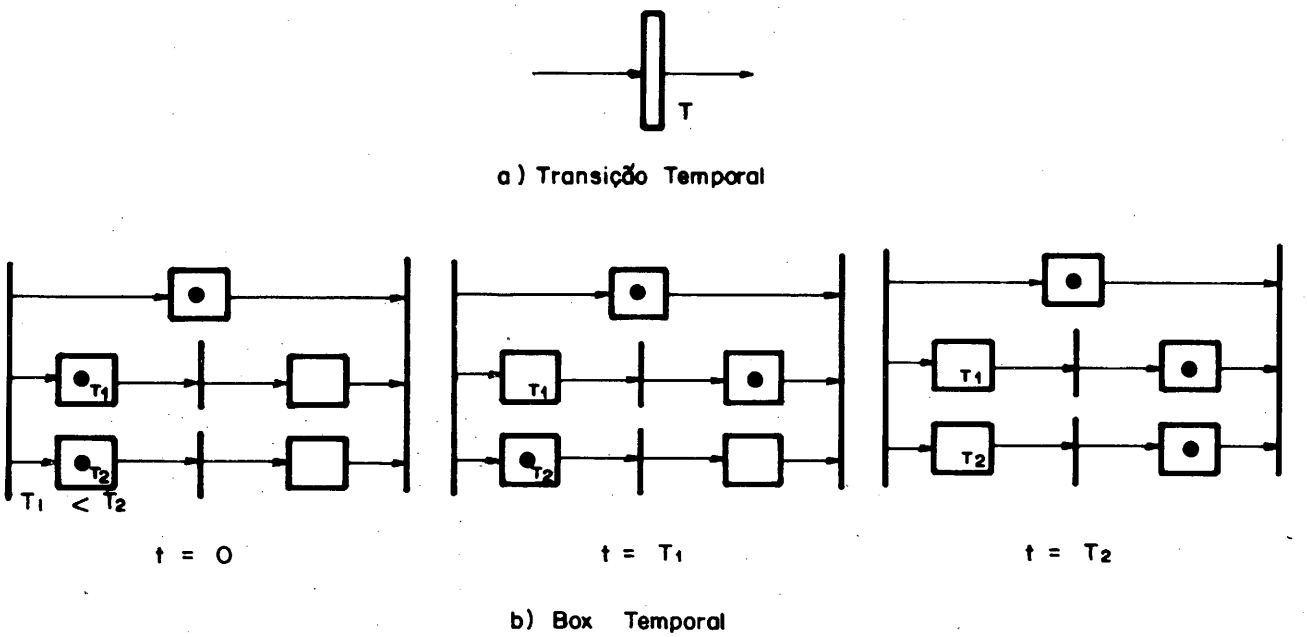


Fig.3. Elementos Temporais

A figura 3. mostra a representação de uma transição temporal e o funcionamento de um box temporizado. Note-se que a introdução de elementos temporais torna mais realístico os modelos de sistemas baseados em MFG.

2.2.3 Definição de macros

Representar um sistema somente por elementos do MFG pode levar a um modelo não muito fácil de ser compreendido. Para facilitar o entendimento de sistemas extensos introduz-se macro-definições que permitem simplificar a representação total do MFG. Algumas destas macro-definições são:

i) "Box de capacidade"

O "box" de capacidade (figura 4) permite ressaltar a capacidade de armazenamento dos elementos do sistema. Este "box" aceita um número especificado de marcas (N), e contribui para habilitar a transição a ele conectada da seguinte forma:

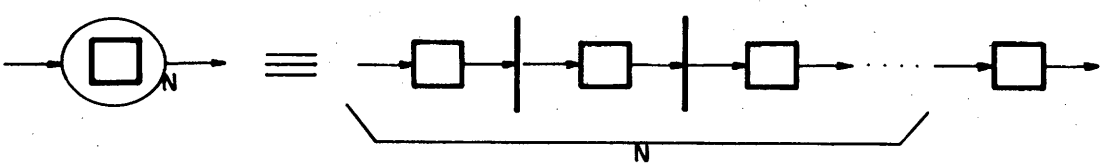


Fig.4. Representação do box de capacidade

– a transição de entrada fica habilitada se o número de marcas no "box" for menor que N;

– a transição de saída fica habilitada se existir pelo menos uma marca no box.

ii) Box" de montagem

O "box" de montagem (figura 5) permite acumular marcas até atingir um valor especificado (N), e então liberar apenas uma marca.

A contribuição deste "box" para as transições conectadas a ele é a seguinte:

– a transição de entrada fica habilitada se o número de marcas no "box" for menor que N;

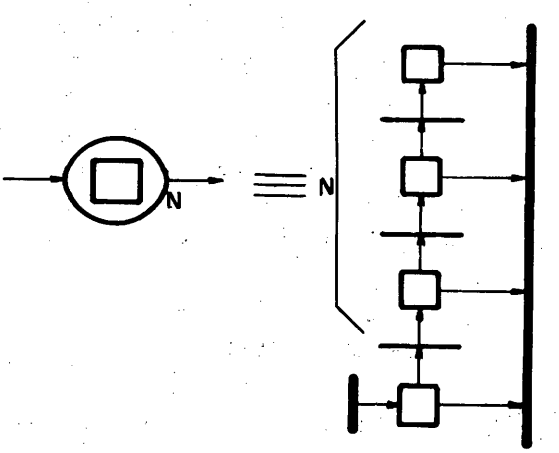


Fig.5. Representação do box de montagem

– a transição de saída fica habilitada se existirem N marcas no box.

iii) "Box" de fornecimento

O "box" de fornecimento (figura 6), recebe um determinado número de marcas (N), liberando uma a cada disparo da transição de saída. Contribui para habilitar as transições conectadas a ele da seguinte forma:

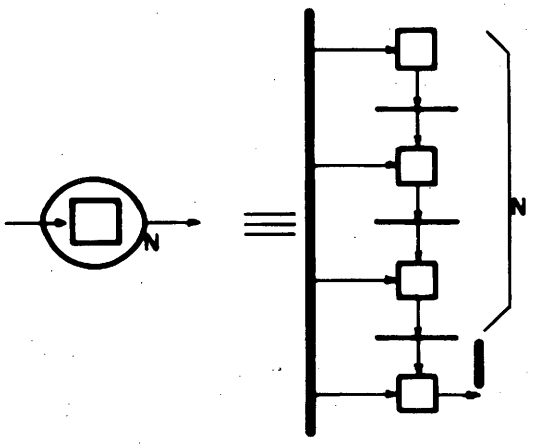


Fig.6. Representação do box de fornecimento

– a transição de entrada fica habilitada se não existirem marcas no "box";

– a transição de saída fica habilitada se existir pelo menos uma marca no "box".

Através do MFG de um sistema de eventos discretos é possível identificar pontos de estrangulamento, isto é, pontos onde possíveis conflitos no fluxo de itens e/ou controle podem ocorrer. Estes pontos são caracterizados pelos "boxes de conflito".

Definição: Um "box" é dito "box de conflito" se este é um "box" de saída de mais de uma transição.

Um outro atributo importante da representação MFG é a identificação de "deadlocks". Se o MFG não contém "boxes de conflito", então ocorre uma estagnação completa do fluxo de controle ("deadlock"), se e somente se existe um loop no grafo, ou é possível criar um loop invertendo alguns dos arcos segundo o teorema da inversão(Masuda, Hasegawa, 1980).

Se o MFG contém "boxes de conflito", o "deadlock" pode ser identificado se o sistema não consegue completar um ciclo de produção (retornar ao estado de marcas inicial) ou se uma transição não conserva o número de marcas, isto é, tem zero marcas no total, ou todas as marcas concentradas em um determinado loop.

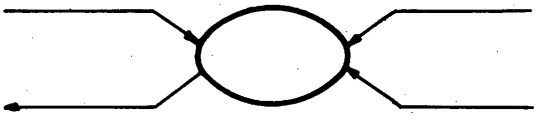
2.3 O Program Flow Schema (PFS)

O "Program Flow Schema" (PFS) (Miyagi, 1988) introduz uma metodologia de projeto ao modelamento de sistemas de eventos discretos em MFG. Esta metodologia pressupõe vários níveis de abstração no processo de modelamento que podem ser conectados de forma estruturada até atingir o nível mais elementar – o nível em que todo o sistema está descrito por um grafo MFG.

O encapsulamento de informações e, principalmente, de decisões de projeto, é monitorado por novos elementos, classificados como : elementos de atividade, distribuição e conexão.

Uma atividade (fig. 7 a) é uma função executada por um subsistema que deve ser descrito em MFG. Portanto, podemos chamar de "nível de atividade" ao nível de abstração em que o sistema é descrito conceitualmente por funções cujo mecanismo de execução está encapsulado na representação PFS.

a) Elemento de Atividade



b) Elemento de Distribuição

Fig.7. Descrição PFS dos principais elementos do sistema

Os elementos de distribuição são elementos controladores do fluxo de itens, em geral elementos de armazenagem (fig. 7 b). Estes elementos não determinam um nível de abstração mas estão diretamente associados aos elementos de atividade por arcos orientados (elementos de conexão).

Usando PFS, um projetista pode iniciar o modelamento de um sistema de eventos discretos por uma descrição conceitual, no"nível de atividade", e, a partir do grafo produzido (uma representação mais abstrata do MFG), passar para níveis inferiores (detalhando o funcionamento das atividades). Este atributo é explorado pela arquitetura do AIPUSP.

Na seção seguinte mostraremos um exemplo de como o esquema PFS poderia ser utilizado para modelar um processo de "disassembly" e posterior empacotamento das partes.

3. UM EXEMPLO DE SISTEMA DE EVENTOS DISCRETOS MODELADO EM PFS.

A figura 8 mostra um exemplo simples mais realístico de um sistema de produção, composto por dois braços mecânicos (para carga e descarga de material); um procedimento de "disassembly"; um "box" de estoque intermediário – que armazena os ítens saídos do processo de "disassembly"; um segundo procedimento, envolvendo o transporte de itens em uma esteira rolante até o empacotamento, que consiste em receber uma embalagem de um subsistema externo, empacotar cinco itens e enviá-los para o estoque final.

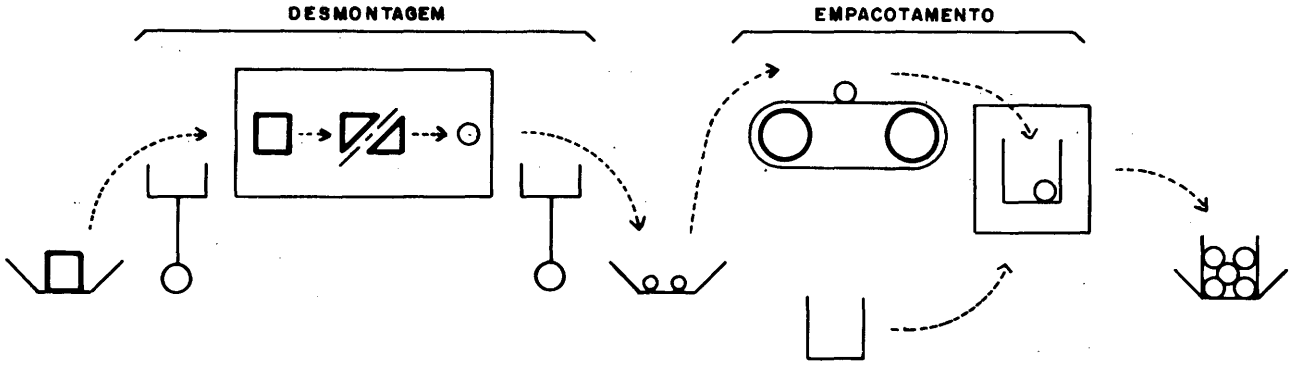


Fig.8. Exemplo de um sistema de produção

Uma descrição inicial do sistema acima, utilizando os níveis de abstração do PFS é mostrado na figura 9. O estoque inicial, intermediário e final são representados por círculos ("boxes" de armazenagem) com capacidade de um, dois e um item respectivamente associados a *buf1*, *buf2* e *buf3* na figura. Entre estes "boxes" de armazenagem estão os procedimentos de "disassembly" e de empacotamento, que, neste nível, são representados pelas atividades */proc1* e */proc2*, respectivamente.

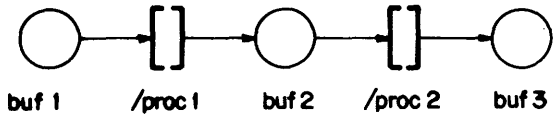


Fig.9. Representação MFG do processo da Fig.8

O próximo passo no modelamento do sistema seria a definição da representação MFG das atividades */proc1* e */proc2*. Note-se que a estratégia de modelamento destas atividades é a mesma utilizada para produzir o modelo da figura 9. Em outras palavras, um sistema de produção pode ser considerado como uma atividade (chamada atividade principal ou "main") cuja representação MFG é definida em função de outras atividades. Estas, por seu turno, serão definidas da mesma forma que a atividade principal por uma aplicação recursiva do mesmo esquema.

As figuras 10 e 11 mostram os níveis de abstração deste pequeno projeto juntamente com a sua descrição na linguagem de controle PFS/MFG.

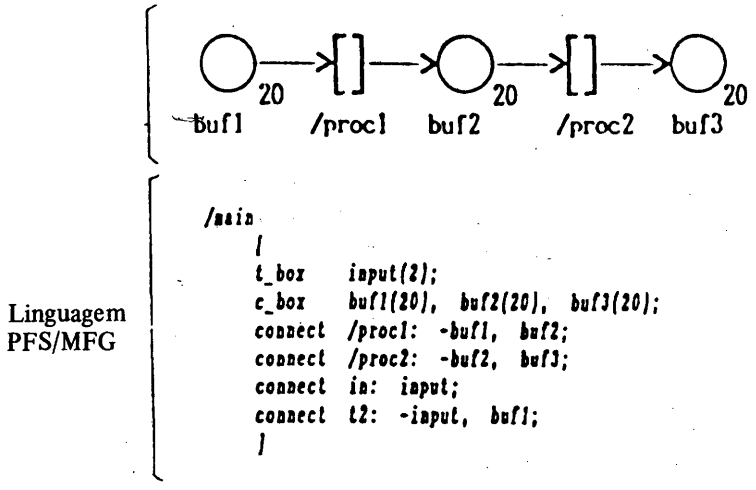


Fig.10. Representação PFS/MFG do exemplo da Fig.8 (nível 1)

4. O SISTEMA AIPUSP

4.1 Histórico

Como a maioria dos sistemas destinados a automação de projetos de sistemas de produção, o AIPUSP foi concebido com o objetivo de compor um ambiente integrado de desenvolvimento a partir de ferramentas clássicas (simuladores procedurais, editores de redes de Petri, etc.), acoplando estas ferramentas a outros módulos inteligentes, que supririam a lacuna deixada pela ausência de ferramentas de análise, da incorporação da experiência dos projetistas, de conhecimento específico do tipo de produto ou do tipo de fábrica que se deseja modelar.

Os primeiros resultados (primeira versão do protótipo) colocaram de pronto a questão da restrição o escopo do projeto para, por exemplo, sistemas flexíveis de manufatura, e introdução de conhecimento específico sobre o produto final (Fisher, 1986), (Smith et alii., 1986) ou da insistência na busca por uma metodologia de desenvolvimento para sistemas discretos de produção em geral. As especificações do sistema, apresentadas na seção seguinte, mostram uma clara opção pela segunda alternativa.

4.2 Especificações do sistema (Lucena, 1989)

O AIPUSP pode ser visto como uma associação de módulos independentes, todos inteligentes, exceto o pacote de simulação (**) (figura 13). Operacionalmente, o sistema suporta dois tipos de enfoques: do ponto de vista do usuário, e do ponto de vista interno do sistema.

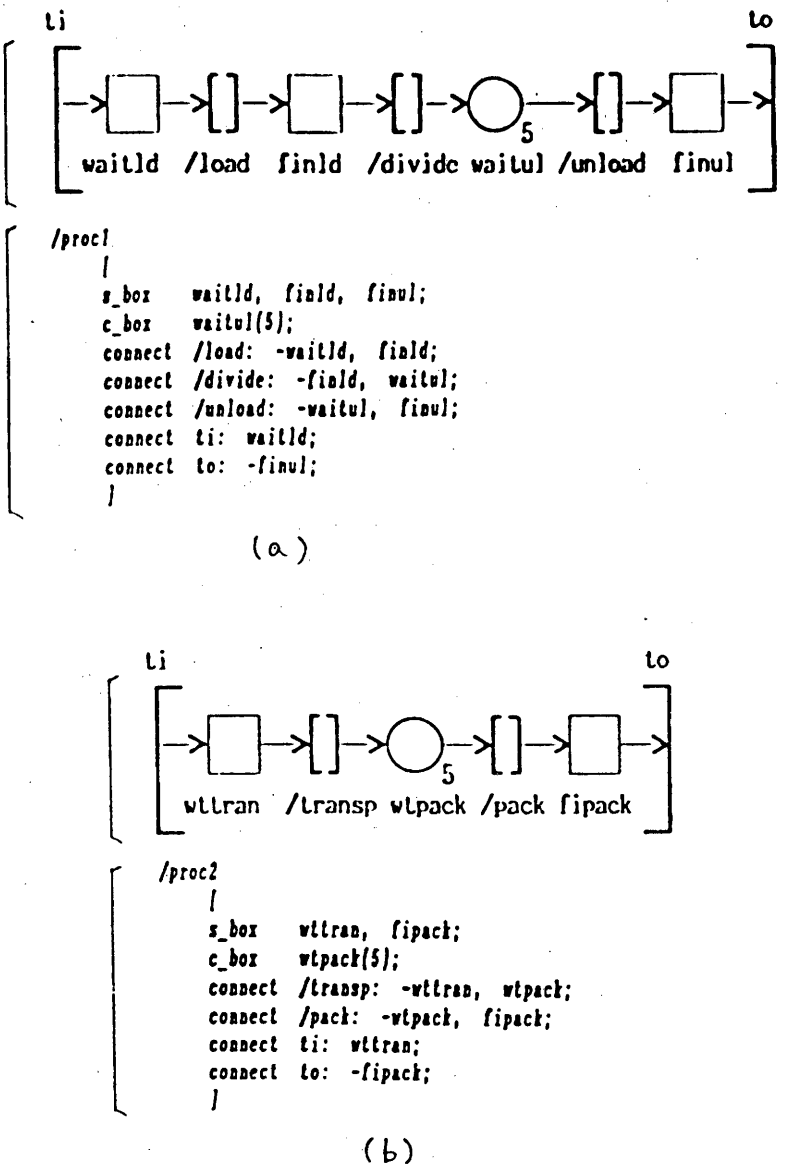


Fig.11. Representação PFS/MFG de /proc1 e/proc2

** O pacote de simulação foi cedido em convênio, pelo Tokyo Institute of Technology e é de autoria do Prof. Huoji Takahashi

```
/main
    [
    t_box    input(2);
    c_box    buf1(20), buf2(20), buf3(20);
    connect  /proc1: -buf1, buf2;
    connect  /proc2: -buf2, buf3;
    connect  in: input;
    connect  t2: -input, buf1;
    /proc1
        [
        s_box    waitld, finld, finul;
        c_box    waitul(5);
        connect  /load: -waitld, finld;
        connect  /divide: -finld, waitul;
        connect  /unload: -waitul, finul;
        connect  ti: waitld;
        connect  to: -finul;
        /load
            [
            s_box    robot1;
            t_box    grasp1(1), move1(2), place1(1);
            connect  stmove: -grasp1, move1;
            connect  stplace: -move1, place1;
            connect  ti: robot1, grasp1;
            connect  to: -robot1, -place1;
            gate  ti: -/proc1/finld;
            ]
        /divide
            [
            s_box    waitdv, wtpro1, wtpro2, tobuf;
            t_box    div(2), proc(2);
            d_box    count(2);
            connect  stdv: -waitdv, div;
            connect  findv: -div, wtpro1, wtpro2;
            connect  stpro1: -wtpro1, proc;
            connect  stpro2: -wtpro2, proc;
            connect  finpro: -proc, tobuf;
            connect  ti: count, waitdv;
            connect  to: -count, -tobuf;
            ]
        /unload
            [
            s_box    robot2;
            t_box    grasp2(1), move2(2), place2(1);
            connect  stmove: -grasp2, move2;
            connect  stplace: -move2, place2;
            connect  ti: robot2, grasp2;
            connect  to: -robot2, -place2;
            gate  ti: -/proc1/finul;
            ]
        ]
/proc2
    [
    s_box    wttran, fipack;
    c_box    wtpack(5);
    connect  /transp: -wttran, wtpack;
    connect  /pack: -wtpack, fipack;
    connect  ti: wttran;
    connect  to: -fipack;
    /transp
        [
        s_box    sttran, fintra;
        t_box    trans1(2), trans2(2), trans3(2);
        c_box    conv(3);
        connect  t1: -sttran, trans1;
        connect  t2: -sttran, trans2;
        connect  t3: -sttran, trans3;
        connect  t4: -trans1, fintra;
        connect  t5: -trans2, fintra;
        connect  t6: -trans3, fintra;
        connect  ti: conv, sttran;
        connect  to: -conv, -fintra;
        ]
    /pack
        [
        s_box    wtmove, posi1, posi2, posi3,
                 posi4, posi5;
        t_box    move1(3), move2(2), move3(2),
                 move4(1), move5(1), pac(2);
        g_box    count(5);
        connect  stmv1: -wtmove, move1;
        connect  stmv2: -wtmove, move2;
        connect  stmv3: -wtmove, move3;
        connect  stmv4: -wtmove, move4;
        connect  stmv5: -wtmove, move5;
        connect  finmv1: -move1, posi1;
        connect  finmv2: -move2, posi2;
        connect  finmv3: -move3, posi3;
        connect  finmv4: -move4, posi4;
        connect  finmv5: -move5, posi5;
        connect  stpack: -posi1, -posi2, -posi3,
                         -posi4, -posi5, pac;
        connect  ti: count, wtmove;
        connect  to: -count, -pac;
        gate  stmv1: -posi1;
        gate  stmv2: -posi2;
        gate  stmv3: -posi3;
        gate  stmv4: -posi4;
        gate  stmv5: -posi5;
        ]
    ]
]
```

Fig.12. Descrição PFS/MFG (completa) do exemplo da Fig.8

Do ponto de vista do usuário a linguagem do sistema é o MFG, mostrado na seção 2. Isto implica que o usuário pode editar, modificar, consultar e simular novos e antigos projetos de sistemas de produção utilizando a linguagem gráfica do MFG. Uma vez validado o modelo, o sistema pode ser sintetizado em uma linguagem de controle, o PFS/MFG que é a linguagem interna do sistema.

A razão de se escolher o PFS/MFG como linguagem interna do sistema é, em primeiro lugar, a economia de memória que se obtém ao construir o banco de dados que armazena antigos

projetos; em segundo lugar esta linguagem guarda com exatidão os níveis de abstração especificados no desevolvimento do modelo, o que garante a recuperação das decisões de projeto. Estes atributos facilitam enormemente o controle de versões e a reutilização de "designs".

Naturalmente, para suportar dois tipos distintos de visões o sistema tem que ser suficientemente versátil na tradução de uma linguagem para outra. Este trabalho é feito por dois tradutores automáticos de linguagens: o primeiro traduz da linguagem gráfica para o PFS/MFG e situa-se na Interface; o segundo é uma ferramenta de uso comum entre os módulos, e traduz do PFS/MFG para a linguagem gráfica.

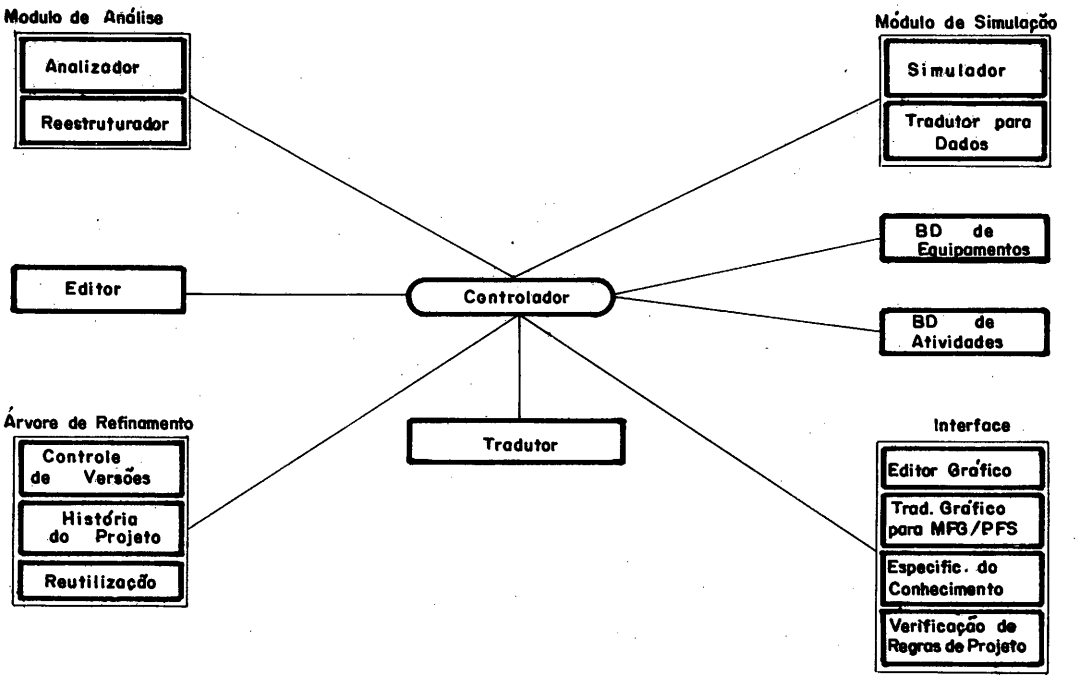


Fig.13. Diagrama de módulos do AIPUSP

Os quatro módulos básicos do AIPUSP, Interface, Árvore de Refinamento, Pacote de simulação e Analisador de projetos podem ser vistos na figura 13.

Além dos módulos básicos existem ainda ferramentas de uso comum entre os módulos, como o tradutor de PFS/MFG para representação gráfica, já mencionado, e um editor de propósito geral, – utilizado para manutenção do sistema. Através deste editor é possível ter acesso e modificar versões de projeto na própria linguagem PFS/MFG, editar documentação de projeto, comentários etc. Existem ainda dois bancos de dados de uso comum: o banco de dados de equipamentos, que guarda informações sobre a indústria específica que se deseja modelar, sobre os equipamentos utilizados nesta indústria, seu custo operacional, taxa de produtividade, etc.; e o banco de dados de atividades (note-se que cada projeto pode ser considerado uma atividade), que guarda antigos projetos com sua respectiva história de decisões e níveis de abstração. Este banco de dados é usado para a reutilização de modelos de sistemas.

Faremos a seguir uma descrição suscinta dos módulos básicos.

4.2.1 Interface

A função básica da Interface é manter um alto nível de interação com o projetista. O editor gráfico permite que este possa introduzir modelos de sistemas discretos diretamente na representação MFG. O modelo é então traduzido para a linguagem PFS/MFG. Neste processo de tradução algumas regras de projeto são verificadas (quadro I).

O banco de regras de conhecimento específico é utilizado para prestar assistência ("help") ao projetista em tempo de execução. Neste banco de regras estão contidas informações, por exemplo, sobre o tipo de produto ou firma para a qual é dedicado o projeto de sistema.

QUADRO I

Algumas regras do projeto

R1 = um elemento de qualquer tipo, isolado (sem conexões), não deve existir. Ex: a linguagem MFG/PFS permite definir um u_box, sendo que as conexões são feitas posteriormente. A ausência destas conexões no programa caracteriza um elemento isolado.
R2 = um elemento com aridade incorreta não deve existir. Ex: a aridade mínima de um u_box é dois, sendo um arco de entrada e outro de saída. Não podem existir arcos de entrada sem seus correspondentes de saída e vice-versa.
R3 = Verificar consistência com o nível superior. Ex.: a definição de um u_box, em um nível, implica na sua existência nos níveis inferiores.
R4 = atividades em cascata devem conter um elemento de armazenamento entre eles. Esta regra é relativa ao domínio de aplicação do sistema.
R5 = verificar consistência em relação ao uso dos equipamentos. Ex.: para um determinada operação, poderia ser escolhido um equipamento e um determinado modelo de controle que não é igual ao especificado na rede.

4.2.2 Árvore de Refinamento

A árvore de refinamento é uma árvore "AND/OR" que guarda todos as decisões de projeto associadas às suas respectivas versões. Uma versão de projeto é um caminho ligando a raiz da árvore (uma atividade principal) a uma folha, isto é, a um nível de detalhamento onde o projeto pode ser inteiramente expresso pela representação MFG.

As atividades pertencentes a um determinado nível de abstração são ligadas por um nó "AND". Os nós "OR" especificam alternativas de projeto, isto é, diferentes escalonamentos e/ou disposições físicas dos elementos. Cada um destes nós ("AND" ou "OR") contém um campo contendo o código PFS/MFG correspondente ao nível de abstração (e a versão) do projeto, e um campo contendo comentários e documentação relacionada com o código – descrição funcional, especificação formal, etc. Apenas o nó raiz de cada atividade é diferente dos demais. Este nó contém uma representação da atividade que corresponde ao elemento da figura 14, e informações adicionais tais como: lista dos projetos nos quais a atividade foi utilizada, lista de índices mapeando os elementos da representação MFG da atividade a máquinas reais, descritas no banco de dados de equipamentos, etc.

O controle de versões é feito ativando e desativando caminhos que levam do nó principal a um nó terminal ("folha")(**).

** Um nó é dito terminal se o código PFS/MFG associado a ele não contém nenhuma atividade não definida

Além da importância evidente da árvore de refinamento na documentação do projeto e na reutilização de atividades, este módulo desempenha um importante papel de suporte para as ações do núcleo que controla e integra as ações de cada um dos módulos básicos do AIPUSP.

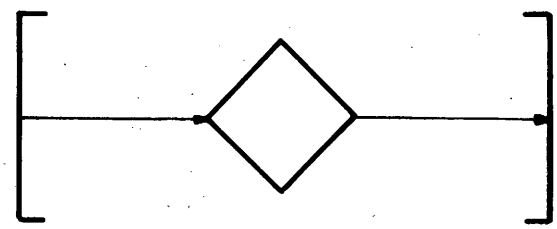


Fig.14. Box de tempo

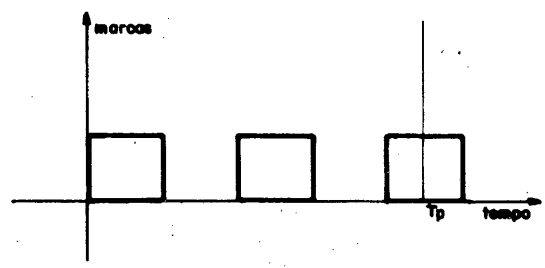


Fig.15. Gráfico (marcas x tempo)

4.2.3 Módulo de Simulação

O módulo de simulação consiste de um compilador e de um simulador de eventos discretos convencional. O compilador aceita código PFS/MFG e gera um programa, já otimizado, que é passado para o simulador. O simulador interpreta cada atividade como um arranjo composto de duas transições e um box de tempo (figura 14). O tempo associado a este box é determinado pelo usuário ou, caso este não deseje estabelecer nenhuma expectativa de tempo, o sistema assumirá o tempo mínimo possível. Deste modo, o simulador pode processar inclusive modelos não-terminais (isto é, cujas atividades ainda não foram completamente definidas).

Os resultados da simulação fornecem informação sobre a existência de "deadlocks", sobre o tempo necessário para se completar um ciclo de produção, sobre a existência de pontos de estrangulamento (que não necessariamente levarão a uma parada total do sistema) e finalmente sobre a taxa de ocupação dos "boxes" (número de marcas processadas em um ciclo de produção). Estes dados serão utilizados pelo analisador.

4.2.4 O Analisador

O analisador é um sistema especialista cujos objetivos são,

i) determinar possíveis razões para uma falha do sistema tais como "deadlocks" ou transgressões das restrições (especificações) previamente estabelecidas pelo usuário, como por exemplo, máxima produtividade, mínimo custo, etc;

ii) sugerir possíveis modificações que poderiam ser implementadas no modelo original para atingir os objetivos iniciais.

O processo de análise se inicia por uma avaliação da eficiência de cada elemento. Por exemplo, a eficiência de um "box" de tempo pode ser calculada pela relação,

$$E = (NK + T_p - Tf) / T_p$$

onde N é o número de marcas processadas, K é o tempo de processamento, Tp é tempo decorrido do início até o ponto em que a simulação foi interrompida, e Tf é o tempo em que ocorreu o último disparo de transição – supondo-se é claro que todas as variáveis temporais estejam normalizadas com relação ao tempo de um ciclo de produção. Isto significa que se considera o sistema "em regime", somente após este ter atingido um ciclo de produção.

Outras fórmulas similares podem ser escritas para os "boxes" de montagem, desmontagem, etc., as quais também dependem do número de marcas processadas. Porém para se obter uma análise mais realística, se utiliza heurísticas extraídas da experiência de especialistas no modelamento de sistemas de eventos discretos. Algumas destas regras heurísticas são também utilizadas para gerar sugestões sobre como melhorar o "design" do sistema. Algumas destas regras se encontram no quadro II a seguir.

QUADRO II

R1 = equipamentos com a mesma funcionalidade podem ser substituídos uns pelos outros, ressalvadas condicionantes de produtividade, custo, espaço físico, entre outras.

R2 = equipamentos sub-utilizados podem ser substituídos por outros mais simples. Ex.: uma esteira com detetores de partes no início e no fim pode ser substituída por uma esteira com detetor apenas no fim, se o outro detetor não está sendo utilizado.

R3 = os elementos de armazenamento podem ser simplificados, se sua capacidade máxima não é atingida. Ex.: se for especificado um elemento de armazenamento com capacidade N e a simulação revelar que somente M<N partes são efetivamente armazenadas então este pode ser substituído por outro elemento com capacidade M.

R4 = um equipamento pode ser duplicado se alimenta outro cuja eficiência não atingiu 100%, ressalvadas as condições de armazenamento intermediário e a produtividade desejada, entre outras.

R5 = um equipamento cujo elemento de armazenamento de entrada atinge a capacidade máxima deve ser duplicado ou substituído por outro com tempo de processamento inferior.

R6 = um equipamento cujo elemento de armazenamento de saída não supera a capacidade mínima deve ser duplicado ou substituído por outro com tempo de processamento inferior.

R7 = equipamentos compartilhados são possíveis pontos de estrangulamento e devem ser duplicados ou substituídos por outros, na tentativa de melhorar a eficiência.

4.2.5 Controle e Integração do Ambiente

Todas as ações dos módulos do AIPUSP são coordenadas e integradas por um módulo controlador. Este módulo tem portanto a tarefa de zelar pela observância de alguns passos considerados essenciais para a imposição da metodologia de desenvolvimento adotada, tais como, documentação adequada das decisões de projeto, controle de versões, validação de cada nível antes de prosseguir especificando mais concretamente o modelo, etc.

Existem dois tipos principais de acessos aos módulos:

i) Acesso supervisionado: onde a seqüência de acessos a módulos do sistema é verificado pelo controlador;
ii) Acesso direto: Neste caso o controlador verifica apenas a passagem de parâmetros, isto é, a seqüência de chamada não é importante para a aplicação da metodologia. Este acesso ocorre com as ferramentas de uso comum como o editor, o tradutor de PFS/MFG para a representação gráfica, e os bancos de dados.

```
*****************************************

A I P U S P
Ambiente Inteligente para o Projeto
de Unidades de Sistemas de Producao

LAS/Mecatronica      Depto. de Eng. Mecanica
LSD/FDTE - USP       Depto. de Eng. Eletrica
     Maio 1989          versao A02

Apoio: S!D - Informatica

*****************************************

>> Deseja trabalhar com
   projeto ja' existente ? (S/N) s

>> Entre com o nome do projeto
   de trabalho ja' existente : t02
```

Fig.16. Tela inicial do Sistema AIPUSP

5. APRESENTAÇÃO DOS RESULTADOS (PROTÓTIPO AIPUSP 0.2)

Nesta seção apresentaremos os resultados obtidos com o protótipo 0.2 do AIPUSP. A função deste protótipo foi a de testar a nossa expectativa de integração entre os módulos inteligentes e as ferramentas procedurais, principalmente com o simulador de eventos discretos – transferido de um ambiente convencional construído pelo Tokyo Institute of Technology.

Mostraremos como esta versão pode tratar o problema do modelamento do sistema de desmontagem e empacotamento mostrado na seção 3. O "menu" principal (figura 16) permite escolher a implementação de um projeto novo ou a edição de uma atividade já elaborada – neste caso o sistema deve se preparar para fazer uma consulta ao banco de dados de atividades.

Uma vez feita a escolha de iniciar um novo projeto, o controlador aciona a interface para colocar o editor gráfico (figura 17) à disposição do usuário. A figura 17 mostra também as primitivas gráficas que podem ser utilizadas pelo projetista. A figura 18 mostra o primeiro refinamento da atividade principal ("main") descrita em função das atividades de desmontagem *(/proc1)* e de empacotamento *(/proc2)*. Estas atividades serão tratadas posteriormente da mesma maneira que a atividade principal. Note-se que este modelo inicial pode ser validado ainda nesta fase antes de dar prosseguimento ao projeto.

A figura 19 mostra a atividade principal acompanhada do primeiro refinamento de */proc1*. Esta opção pode ser gerada utilizando o fato de que todo o projeto já elaborado pode ser armazenado de forma hierárquica na árvore de refinamento. Assim é possível ler o código PFS/MFG armazenado nos nós "AND" e mostrá-los todos em uma mesma tela – se houver

espaço disponível. Cada uma destas telas tem um limite máximo de 50 elementos. Esta restrição é mais uma imposição da metodologia do que propriamente uma simples limitação do sistema.

A árvore de refinamento pode ser alterada utilizando o editor de propósito geral mencionado na seção 4. A figura 20 mostra a janela principal do editor onde se vê o código correspondente ao refinamento da atividade principal do projeto. Note-se que é possível acessar os nós "AND" ou "OR" da árvore a partir deste ponto. Portanto é possível seguir qualquer caminho existente na árvore.

É possível também alterar um nó através da janela de edição

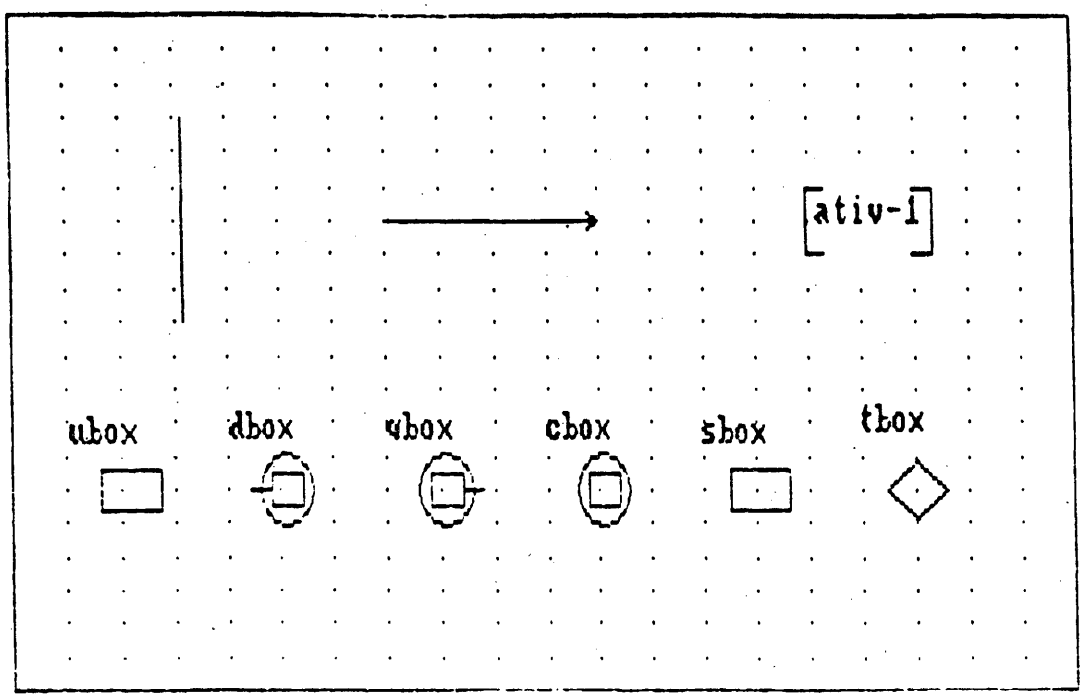


Fig.17. Primitivas básicas do editor

(figura 21). Entretanto, cada vez que uma alteração é introduzida em um nó já existente uma nova alternativa de projeto é gerada, isto é, a alteração deverá fazer parte de um novo caminho ligando a atividade principal a uma "folha".

Terminada a fase de edição o sistema dirige o usuário para a validação do modelo. Em outras palavras, o controlador salva as modificações feitas na árvore de refinamento, encerra a ação deste módulo, e ativa o pacote de simulação. A simulação pode ser feita em "batch", ou pode ser acompanhada através da interface gráfica (figura 22).

Finalmente, os resultados da simulação são transferidos para o analisador que gera a análise de eficiência dos elementos do modelo. Nesta versão inicial a análise de eficiência é feita de forma algoritmica através do cálculo de equações como a mostrada na seção 4.2.4. A razão de não ter sido ainda incluída uma análise baseada em heurísticas como foi previsto na especificação é que o simulador convencional não permite a geração de dados de análise que retratem as decisões de projeto. Portanto, as únicas conclusões possíveis são: a existência ou não de "deadlocks" – neste caso a simulação seria interrompida, a deteção de pontos de estrangulamento – visualmente ou através da análise das taxas de ocupação dos "boxes", e a necessidade de substituição de alguns elementos que apresentem eficiência limitada (figura 23).

Esperamos no futuro poder substituir este simulador por um simulador declarativo no estilo do KBS (Reddy et alii., 1986),o que nos permitirá avaliar a propriedade da disposição espacial dos elementos em uma fábrica, ou o relacionamento do desempenho de uma atividade com decisões anteriores. Esperamos ainda, com esta ferramenta poder implementar um sistema de simulação e análise baseado em "frames".

6. CONCLUSÃO

O AIPUSP tem sido testado em outros problemas-modelo gerados no ambiente convencional pertencente ao Tokyo Institute of Technology apresentando bons resultados. Entretanto, para a próxima versão do protótipo, já iniciada, os autores pretendem introduzir, além do simulador declarativo, mecanismos que aumentem a modularidade do projeto.

Infelizmente, a introdução de modularidade traz também a dificuldade de escolher de maneira apropriada os módulos do projeto, de modo a evitar problemas futuros na composição

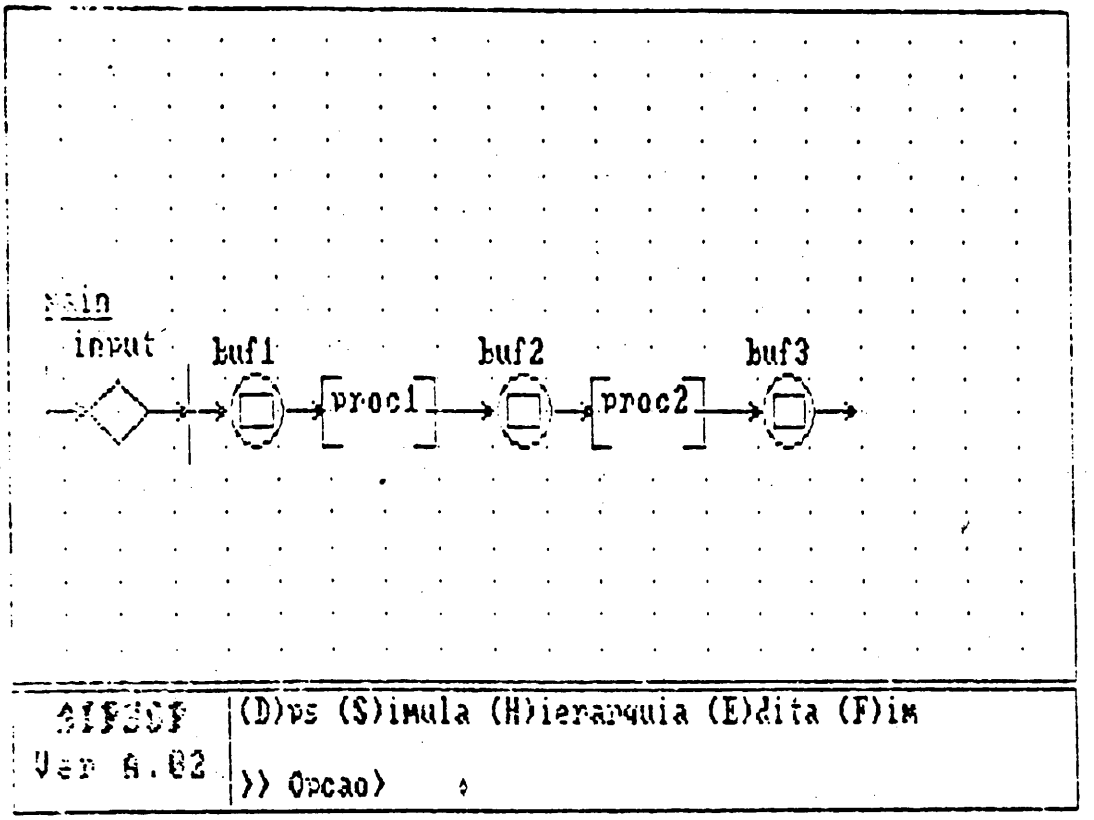


Fig.18. Descrição gráfica do exemplo da Fig.8

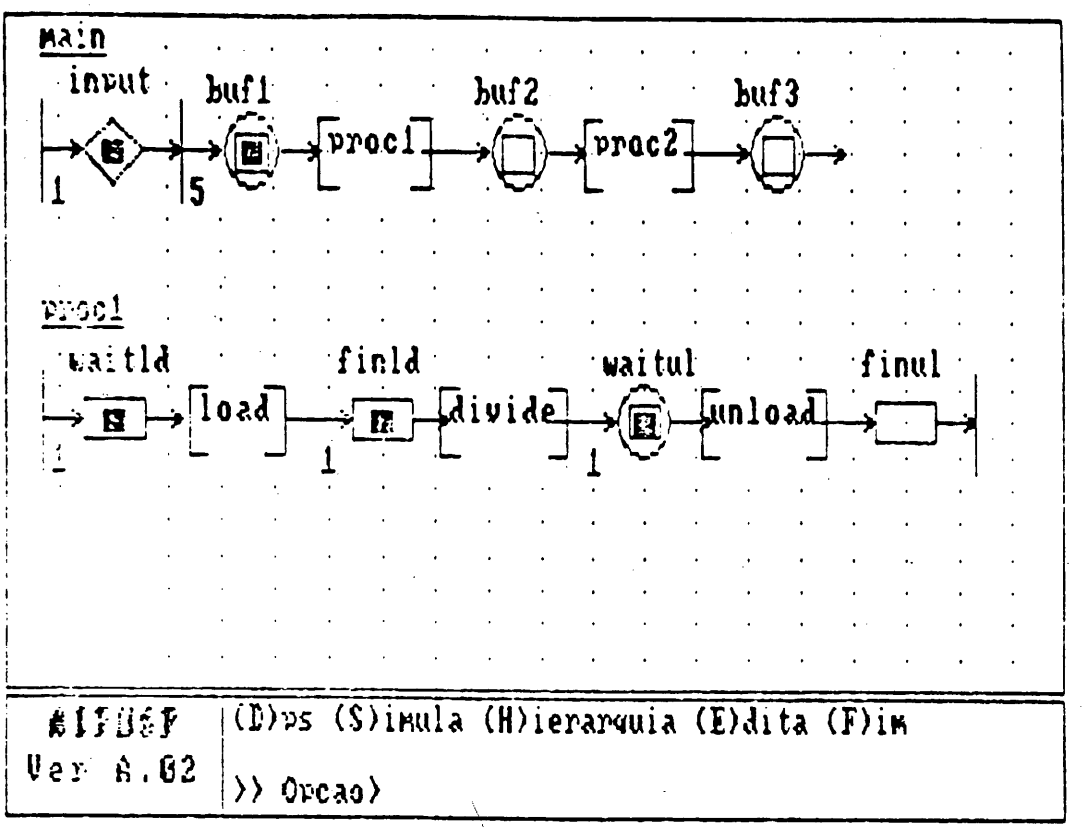


Fig.19. Primeiro nível de refinamento do exemplo.

destes módulos. Os aspectos de sincronização e concorrência que aparecem em sistemas de produção, notadamente em células flexíveis de manufatura, tornam a modularização desse tipo de projeto um objetivo muito difícil de alcançar.

Por outro lado, os autores acreditam que a formalização da metodologia empregada pode trazer novas perspectivas para a implementação do AIPUSP. Atualmente, os autores estudam a possibilidade da utilização de especificações formais baseadas em semântica de ações (Broy, 1988) para tratar os problemas de concorrência. O sucesso parcial obtido com a interface do AIPUSP, para esconder a complexidade do funcionamento interno do sistema, é alentador no sentido de nos permitir pensar em um sistema mais complexo que seja por outro lado acessível aos projetistas de sistemas.

AGRADECIMENTOS

Os autores gostariam de agradecer ao Prof. Kensuke Hasegawa e ao Prof. Paulo Eigi Miyagi pelas discussões mantidas durante a fase de elaboração deste projeto. A avaliação e o incentivo deles tiveram um papel importante para a concentração de esforços no AIPUSP. Agradecemos também ao Prof. Takahashi, autor do simulador convencional utilizado. Finalmente agradecemos aos antigos participantes da equipe do projeto, Marcos R. P. Barreto, Jaime Sichman e João Batista Camardo Jr., pela participação nas discussões iniciais que levaram ao projeto atual.

REFERÊNCIAS

BROWN, D.C. and CHANDRASEKARAN, B.: "Expert Systems for a Class of Mechanical Design Activity". In Proceedings of International Federation of Information Processing Society. Working Conference on Knowledge Engineering in Computer Aided Design; North-Holland, Amsterdam, 1984.

HASEGAWA, KENSUKE et al., "Simulation of Discrete Production Systems Based on Mark Flow Graphs"; System Science, vol. 13, no. 1-2, Poland, 1987.

MASUDA, RYOSHI E HASEGAWA, KENSUKE; "Mark Flow Graphs and its Application to Complex Sequencial Control System"; Proc. 13th Int. Conference on System Science, Hawaii, 1980.

MIYAGI, PAULO E.; "Control System Design, Programming and Implementation for Discrete Event Production Systems by Using Mark Flow Graphs", tese de doutoramento, Tokyo Institute of Technology, 1988.

MOSTOW, JACK; "Toward Better Models of the Design Process", The A.I. Magazine, Spring 1985.

HASEGAWA, KENSUKE AND TAKAHASHI, KUOJI: "MFG (Mark Flow Graphs)-Based Programmable Controller and its Application to FMS"; Proc. of 3rd Int. Conference on System Engineering, Ohio, 1984.

BRUNO, GIORGIO; ELIA, A., and LAFACE, P.: "A Rule-Based System to Schedule Production"; IEEE Computer, July 1986.

POTTS, COLIN and BRUNS, GLEEN; "Recording the Reasons for Design Decisions", IEEE, 1988.

FISHER, EDWARD: "An AI-Based Methodology for Factory Design"; AI Magazine, Fall 1986.

SMITH, S.; FOX, M. AND OW, PENG: "Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems"; AI Magazine, Fall 1986.

REDDY, Y.V.R.; FOX, M.; HUSAIN, N. and MCROBERTS, M.: "The Knowledge-Based Simulation System"; IEEE Software, March 1986.

LUCENA, C. et al.: "The Specification of a Knowledge-Based Environment for the Design of Production Systems"; to appear on Proc. of 6th Symposium on Information Control Problems in Manufacturing Technology"; Madrid, September 1989.

BROY, MANFRED (1988). "Distributed, Concurrent & Reative Systems, State of Art Seminar on Formal Description of Programming Concepts". Petrópolis, Brazil.

```
Conteudo de uma atividade
     Atividade :  main                ATIVA
  Atividade pai :  root
        Filhos :  [proc1,proc2]

  Proxima Atividade        Descricao
  ▶ Filho                  Definicao da atividade
    Pai                    main
    Irmao - AND
    Outro - OR
                                                 00001:001
                  Programa PFS/MFG
  Edita           t_box   input( 2 );
                  c_box   buf1( 20 ), buf2( 20 ), buf3( 20 );
                  connect /proc1: -buf1, buf2;
  Final           connect /proc2: -buf2, buf3;
                  connect in      : input;
                  connect t2      : -input, buf1;

  Move
                                                 00001:001
```

Fig.20. Árvore de Refinamentos

```
>> Atividade :          Sub-atividades :
     main
>> Filiacao :
     root                (*) - Criacao de novo no'
                         ( ) - Correcao do conteudo        Retorna
  Descricao
  Definicao da atividade
  main                                                     Abandona
                                                 00001:001
  Programa PFS/MFG
  t_box   input( 2 );
  c_box   buf1( 20 ), buf2( 20 ), buf3( 20 );
  connect /proc1: -buf1, buf2;
  connect /proc2: -buf2, buf3;
  connect in      : input;
  connect t2      : -input, buf1;

                                                 00001:001
```

Fig.21. Editor da Árvore de Refinamento

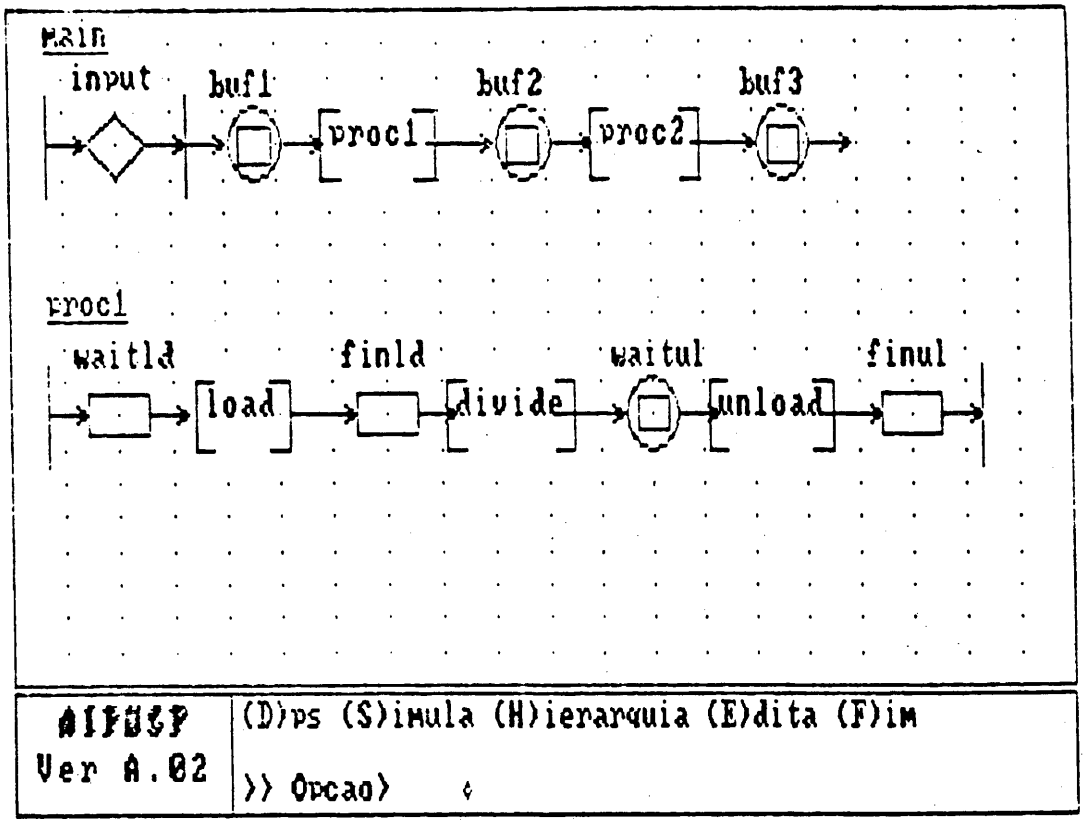


Fig.22. Acompanhamento da simulação com o Editor Gráfico

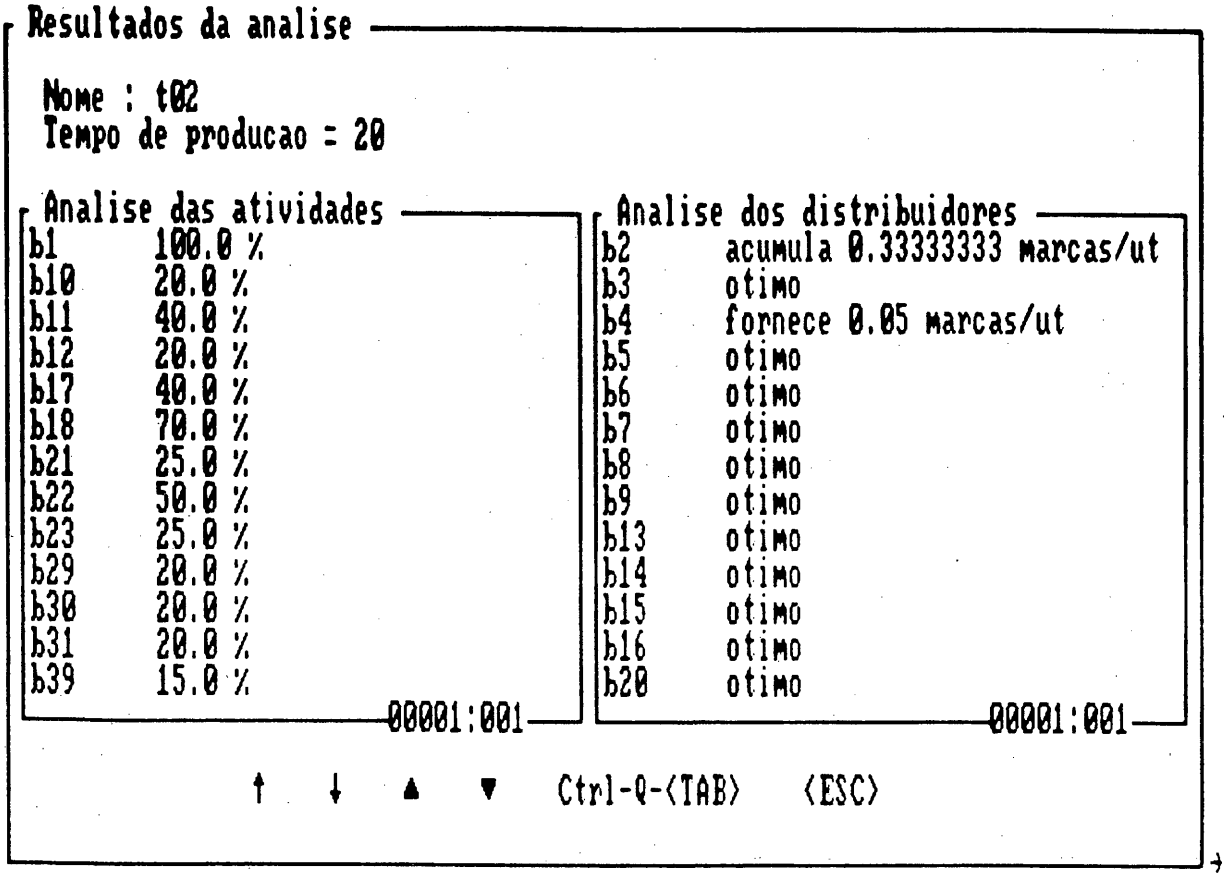


Fig.23. Resultados da Análise