ESTE EXEMPLAR CORRESPONDE A REDAÇÃO FINAL DA TESE DEFENDIDA POR Luís Clovis Lima Viana E APROVADA PELA COMISSÃO JULGADORA EM 22/02/2000.

ORIENTADOR

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

Avaliação de Arquiteturas de Comando: Aplicação a um Sistema Automatizado de Produção Industrial

Autor : **Luís Clovis Lima Viana**
Orientador: **João Maurício Rosário**

38/00

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

Avaliação de Arquiteturas de Comando: Aplicação a um Sistema Automatizado de Produção Industrial

Autor : **Luís Clovis Lima Viana**
Orientador: **João Maurício Rosário**

Curso: Engenharia Mecânica.
Área de concentração: Mecânica dos Sólidos e Projeto Mecânico

Dissertação de mestrado apresentada à comissão de Pós Graduação da Faculdade de Engenharia Mecânica, como requisito para obtenção do título de Mestre em Engenharia Mecânica.

Campinas, 2000
S.P. - Brasil

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

V654m

Viana, Luís Clovis Lima
Metodologia para avaliação de desempenho de arquiteturas de comando : aplicação a um sistema automatizado de produção industrial / Luís Clovis Lima Viana.--Campinas, SP: [s.n.], 2000.

Orientador: João Maurício Rosário.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Simulação (Computadores digitais). 2. Redes de petri. 3. Sistemas de comando e controle. 4. Temporizadores automáticos. I. Rosário, João Maurício. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

DISSERTAÇÃO DE MESTRADO

Avaliação de Arquiteturas de Comando: Aplicação a um Sistema Automatizado de Produção Industrial

Autor : **Luís Clovis Lima Viana**
Orientador: **João Maurício Rosário**

Prof. Dr. João Maurício Rosário, Presidente
Faculdade de Engenharia Mecânica - UNICAMP

Prof. Dr. Geraldo Nonato Telles
Faculdade de Engenharia Mecânica - UNICAMP

Prof. Dr. Ricardo Ribeiro Gudwin
Faculdade de Engenharia Elétrica e de Computação - UNICAMP

Campinas, 22 de Fevereiro de 2000

Dedicatória:

Dedico este trabalho a minha família e a Karina.

Agradecimentos

Eu agradeço profundamente às seguintes pessoas, sem as quais este trabalho nunca teria sido possível:

À minha família, pelo amor e generosidade sem limites.

A Karina, por acreditar no homem.

A meu orientador, cujo entusiasmo funcionou sempre como um farol a me guiar.

Ao Sr. Jean-Marc Faure, verdadeiro co-orientador deste trabalho, pela imensa contribuição para meu crescimento pessoal.

A todas as pessoas do Laboratório de Automação Integrada e Robótica da Faculdade de Engenharia Mecânica da UNICAMP, pelo apoio técnico.

A todos os amigos do CESTI de Toulon, França, pela inestimável ajuda num período crucial de minha vida.

Resumo

VIANA, Luís Clovis Lima, *Metodologia para Avaliação de Desempenho de Arquiteturas de Comando: Aplicação a um Sistema Automatizado de Produção Industrial*, Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2000. 119 p. Dissertação (Mestrado).

O projeto de Sistemas Automatizados de Produção requer, entre outras atividades, a especificação de uma arquitetura de comando. Principalmente em sistemas com estrutura de controle-comando bastante distribuída, esta especificação é determinante para todo o restante do ciclo de vida do sistema em questão. Nesses casos, o projetista vê-se face à difícil tarefa de, numa fase inicial de projeto, escolher uma arquitetura de comando capaz de atender a pré-requisitos funcionais como tempos de resposta do sistema. Este trabalho apresenta uma metodologia para validar um modelo de arquitetura de comando distribuída através da construção de modelos de Análise Estruturada e de sua posterior simulação sob a forma de Redes de Petri Coloridas e Temporizadas. Como estudo de caso, esta metodologia será aplicada a uma plataforma didática em automação.

Palavras Chave

Comando Distribuído, Simulação, Redes de Petri Coloridas e Temporizadas, Características Temporais

Abstract

Viana, Luís Clovis Lima, *Methodology for Performance Evaluation of Control Architectures: Application to an Industrial Automated Manufacturing System*, Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2000. 119 p. Dissertação (Mestrado).

The design phase of Automatic Manufacturing Systems comprises many different tasks and among them stands the design of a control architecture. Actually for highly distributed control systems the choice for a specific architecture has great impact on the overall system's life cycle. In such cases the designer has to propose early enough in the system's design phase a control architecture which must fulfill functional requirements such as response time. This work presents a framework for validating distributed control architectures by modelling the system's distributed features with tools from Structured Analysis and further 'translating' its dynamic behavior into Timed Colored Petri Nets models for simulation. A case study is provided as an application in a didactic automatic platform.

Keywords

Distributed Control, Simulation, Timed Colored Petri Nets, Response Time

Índice

Capítulo 1

Introdução

O desenvolvimento de um Sistema Automatizado de Produção (SAP), desde a especificação de seus pré-requisitos funcionais até o começo de sua utilização, é geralmente modelado como uma seqüência de tarefas ou macro-atividades mais ou menos distintas. Por analogia à Engenharia de *Software*, que desempenhou um papel precursor na normalização de tais processos [JAU90], o desenvolvimento de um SAP pode ser representado por um modelo chamado de *ciclo em V* ou *carta de atividades* [VER91].

A *carta de atividades* (Figura 1) representa o processo realizado durante o desenvolvimento de um sistema complexo. Ela comporta várias fases sucessivas, cada uma delas representando um conjunto particular de operações sobre o SAP. As competências requeridas por cada operação são geralmente diferentes, o que coloca em evidência o caráter pluridisciplinar das equipes encarregadas da realização do sistema. De fato, as disciplinas envolvidas são numerosas e possuem seus próprios métodos, ferramentas e modelos dedicados [COU97].

A organização da carta de atividades indica a sucessão das macro-atividades de desenvolvimento do SAP e a disposição em paralelo de algumas destas. Esta organização gera um eixo temporal (horizontal), pois as diferentes operações do modelo são interdependentes; e um eixo de especialização técnica (vertical), pois, tradicionalmente, quanto mais no alto deste modelo está uma determinada atividade, mais abstrato é o problema (não havendo muitas ferramentas de informática para resolvê-lo) e menor é a necessidade de um especialista numa disciplina específica. Desta forma, a carta de atividades compreende três partes principais:

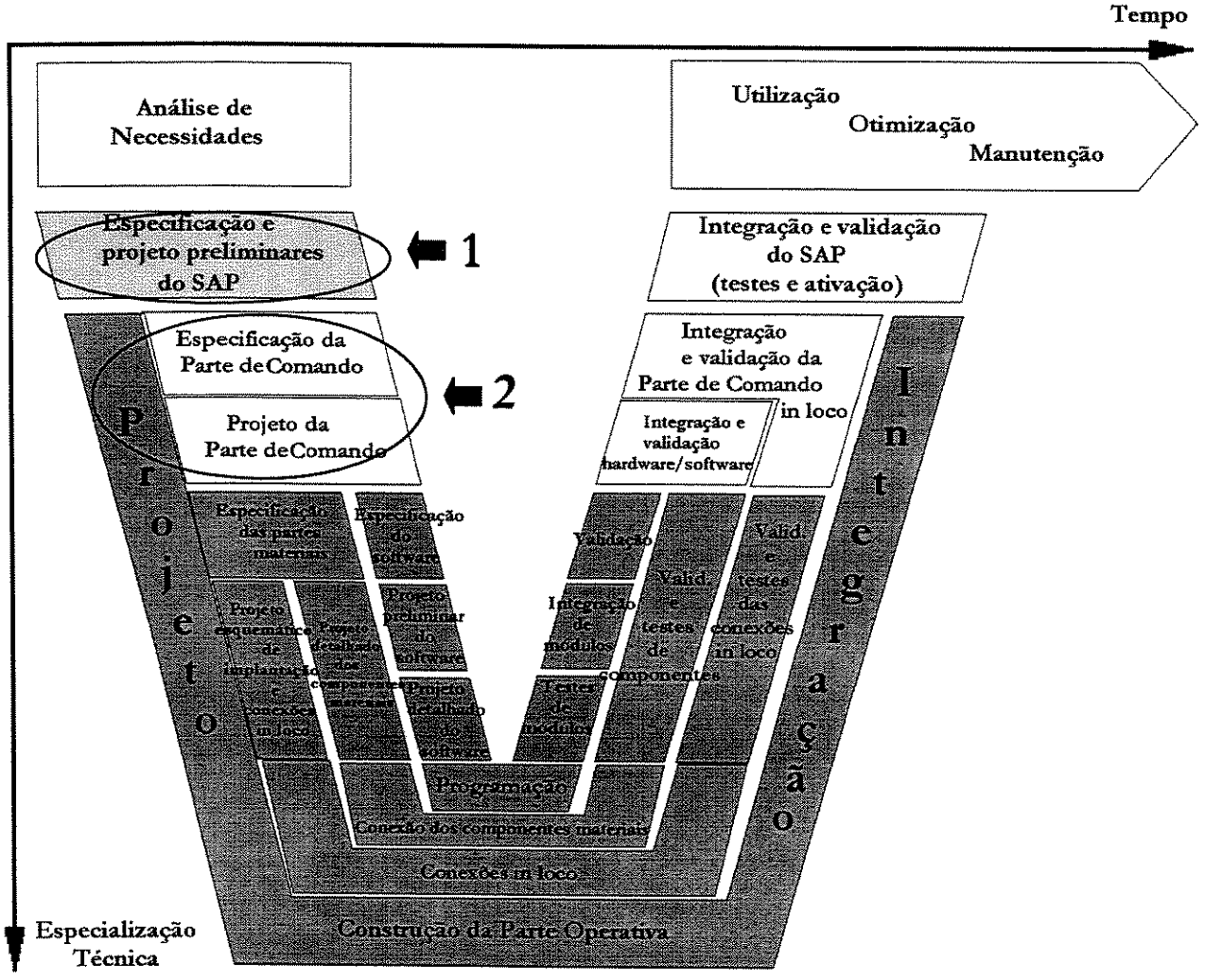


Figura 1. "Ciclo em V" ou carta de atividades do desenvolvimento de um SAP.

• a parte descendente (lado esquerdo) representa as fases de especificação e de projeto do SAP; definem-se então as propriedades do sistema, das mais externas (especificações) às mais internas (de construção), passando por fases de validação, pois cada macro-atividade de projeto produz modelos que devem ser validados antes que se passe à construção do sistema;

• a parte ascendente (lado direito) representa uma seqüência de integrações e testes de subconjuntos cada vez mais complexos (estas integrações e testes são realizados segundo as diretrizes estabelecidas na fase situada no mesmo nível horizontal da parte descendente, e às vezes geram a necessidade de rever uma determinada especificação ou detalhe de projeto); este

período se encerra com uma definição das operações do sistema de acordo com os procedimentos definidos durante a especificação geral do SAP;

- a parte horizontal seguinte corresponde às atividades de operação e de manutenção do SAP, o que completa seu ciclo de vida.

A carta de atividades compreende as fases do desenvolvimento tanto da *parte operativa* quanto da *parte de comando* do SAP. Estas fases são, por sua vez, decompostas em desenvolvimento de *software* e desenvolvimento de *hardware*.

A problemática do ***projeto da arquitetura de comando***, ou seja, a especificação de como as funções de comando do sistema serão executadas pelos diferentes equipamentos disponíveis, é tradicionalmente posicionada nas atividades de desenvolvimento da parte de comando da carta de atividades (região **2** na Figura 1). Na verdade, este posicionamento é relativamente indefinido e depende da situação encontrada originalmente pela equipe de projeto.

Segundo B. Denis [DEN94], esta atividade de projeto, posicionada de tal maneira, no meio do ciclo de desenvolvimento do SAP, é perfeitamente aceitável nos casos em que a escolha de uma arquitetura de comando não é uma atribuição da equipe de projeto: por exemplo, quando trata-se de arquiteturas de comando simples, limitadas, muitas vezes, a um só equipamento de controle; ou no caso da reconfiguração de um sistema de comando cuja repartição de tarefas é predeterminada.

No entanto, visando melhores níveis de flexibilidade e reatividade no meio industrial, o sistema de controle-comando de um SAP é freqüentemente construído de forma distribuída: as partes operativas são comandadas por diversos *controladores lógicos programáveis* (CLP); estes, por sua vez, muitas vezes comandados por um *sistema de supervisão*; as trocas de dados entre as diferentes entidades físicas de comando sendo realizadas por *redes locais*. O trabalho da equipe de projeto, no momento em que deve escolher entre diferentes arquiteturas possíveis, consiste geralmente na avaliação dos seguintes pontos:

- **funcionalidade**, ou seja, a aptidão para realizar o conjunto das funções necessárias à operação;

- **segurança**, em particular a disponibilidade e os comportamentos limites do sistema;
- **características de funcionamento**, particularmente os tempos de resposta oferecidos pelas diferentes arquiteturas e o desempenho temporal global do sistema.

É extremamente difícil avaliar estes pontos através de estudos de documentação técnica, sobretudo analisar os tempos de resposta e os comportamentos limites das arquiteturas possíveis de um sistema distribuído. De forma geral, os fabricantes de equipamentos oferecem documentação completa sobre os componentes de seus sistemas de controle se considerados individualmente, mas não dispõem de informações sobre a interação de diferentes componentes e parâmetros de influência.

Assim, para SAPs com estruturas de comando fortemente distribuídas, a escolha de uma arquitetura de comando adequada torna-se crucial para todas as atividades seguintes de seu ciclo de vida. Em tais casos, o projeto da arquitetura de comando se situaria antes dos ciclos tradicionais: mais precisamente, durante o **projeto preliminar do SAP** (região 1 na Figura 1).

No entanto, posicionado desta forma, o projeto da arquitetura de comando coloca a equipe de projeto do SAP diante do seguinte paradoxo: a partir da análise de necessidades e antes de passar às fases de projeto propriamente dito do SAP, ter que propor uma arquitetura de comando teoricamente capaz de satisfazer os pré-requisitos funcionais do sistema e com um grau de detalhamento suficiente para assegurar a seqüência das macro-atividades seguintes do desenvolvimento do SAP.

Este cenário, cada vez mais comum devido à complexidade crescente dos sistemas automatizados, gera a necessidade de metodologias capazes de estruturar e formalizar o projeto da arquitetura de comando. Assim, uma metodologia de projeto de arquitetura deveria fornecer à equipe de projeto um conjunto de diretrizes e ferramentas para:

- aumentar a produtividade desta atividade, ou seja, minimizar o tempo despendido na avaliação de soluções de arquitetura de comando, aumentando assim a reatividade perante um mercado competitivo;
- facilitar a transferência de conhecimentos provenientes de projetos anteriores, através da homogeneização dos modelos construídos;

- aumentar a qualidade das arquiteturas concebidas, ao levar em conta os diferentes pré-requisitos do sistema em questão, tais como processamento lógico de dados, seu comportamento dinâmico e as propriedades dos equipamentos envolvidos.

A noção de arquitetura de comando já foi abordada em outros trabalhos [JAU90]. Entre eles, as citações mais freqüentes são feitas à metodologia conhecida como *SA-RT* (*Structured Analysis Real Time*), proposta por Ward e Mellor [WAR86] e enriquecida por Hatley e Pirbhai [HAT91]. O SA-RT, originalmente desenvolvido para o projeto de sistemas de informática de tempo real embarcados, trouxe um grande avanço ao estruturar e formalizar a dinâmica do comando de um sistema, em que *funções lógicas* podem ser ativadas, desativadas ou sincronizadas por meio de *funções de controle*. No entanto, esta metodologia explora apenas muito vagamente os aspectos materiais da arquitetura de comando, como, por exemplo, os reflexos da escolha de uma determinada repartição de funções sobre o conjunto dos equipamentos.

O presente trabalho apresenta uma metodologia que se utiliza de diferentes formalismos para modelar tanto os aspectos lógicos/estáticos de uma arquitetura de comando como seu comportamento dinâmico. É dada ênfase a este último ponto, por ser este o aspecto de mais difícil avaliação no projeto de uma arquitetura de comando. O objetivo desta macro-atividade consistiria, assim, na geração de um modelo de arquitetura de comando **validado**, sobretudo no que diz respeito aos **tempos de resposta do sistema**, antes que se passe às fases seguintes da carta de atividades do SAP, ou seja, as fases de desenvolvimento mais detalhado de software e hardware de sua parte de comando. Para isso, este processo vai desde a construção de modelos apropriados a sistemas distribuídos até a geração de modelos dinâmicos passíveis de serem avaliados por simulação, passando pelas técnicas e paradigmas utilizados na simulação de sistemas de informática e de redes de comunicação.

Capítulo 2

Projeto de um Sistema de Comando Distribuído

O projeto da parte de comando de um SAP vai da especificação das operações necessárias, baseadas nos pré-requisitos funcionais do sistema, até a escolha de uma estrutura material adaptada à execução destas. O principal objetivo desta macro-atividade é determinar quais equipamentos utilizar e, sobretudo, **como as diferentes tarefas e pré-requisitos funcionais da aplicação em questão serão distribuídas no conjunto de equipamentos disponíveis**. Isso consiste, mais precisamente, em **projetar uma arquitetura de comando** para o sistema.

Visando considerar paralelamente os aspectos funcionais e materiais da arquitetura de comando, o projeto de sistemas distribuídos é freqüentemente modelado segundo três pontos de vista diferentes [MEU97]:

- **arquitetura funcional**, ou seja, os modelos que descrevem as necessidades funcionais da aplicação;
- **arquitetura material**, o conjunto de equipamentos a serem utilizados;
- **arquitetura operacional**, a repartição das funções do sistema sobre os diferentes equipamentos.

Uma **arquitetura operacional validada** é o objetivo final das atividades de especificação e projeto da arquitetura de comando. Ela representa uma arquitetura física capaz de comportar as diferentes funções necessárias a uma determinada aplicação respeitando as restrições impostas pelas especificações do sistema, como a separação geográfica de equipamentos, a capacidade de

armazenamento de dados, etc. Particularmente, no caso de sistemas de tempo real, a validação implica o respeito às restrições temporais ligadas à execução. Para tais aplicações, a avaliação do desempenho do sistema é fundamental pois não se trata apenas de produzir resultados corretos em valor mais principalmente em tempo hábil.

2.1. Arquitetura Funcional

A arquitetura funcional é a representação dos pré-requisitos funcionais da parte de comando de um SAP. Ela é modelada como se a estrutura física da parte de comando fosse constituída apenas de um único equipamento de controle-comando, como, por exemplo, um computador industrial.

A estrutura dos modelos que serão adotados para descrever a arquitetura funcional deriva da abordagem conhecida como ***Análise Estruturada*** [YOU89] [DeM79]. Fundamentalmente, esta abordagem busca definir as especificações de um sistema de forma **gráfica, hierárquica** e **minimamente redundante**. As principais ferramentas da Análise Estruturada destinam-se a modelar: os processamentos de dados efetuados pelo sistema; a seqüência e a dependência temporal destes tratamentos; e as relações entre os diferentes conjuntos de dados. Assim, pode-se representar a arquitetura funcional através dos seguintes tipos de modelos:

1) **Modelos funcionais.** A arquitetura funcional deve representar todos os processos lógicos que os programas de comando executam assim como todos os dados que circulam entre estes processos. Ela deve também definir os tempos de vida dos diferentes tipos de dados, ou seja, se os dados são consumidos depois da transferência entre dois processos ou se é preciso prever o armazenamento para assegurar a perenidade da informação. As principais ferramentas utilizadas para desempenhar essas funções são os *diagramas de fluxo de dados*. Uma das variações desses diagramas é o actigrama (Figura 2), proveniente do método SADT (*Structured Analysis and Design Technique*) [ICAM80].

2) **Modelos comportamentais.** É preciso também definir quando os processos de comando devem ser ativados, ou seja, quais são os eventos necessários à realização de um processo e quais são as regras de sincronização entre eles. Tradicionalmente, os métodos derivados da Análise Estruturada (como SA-RT [WAR86] [HAT91]) utilizam os *diagramas*

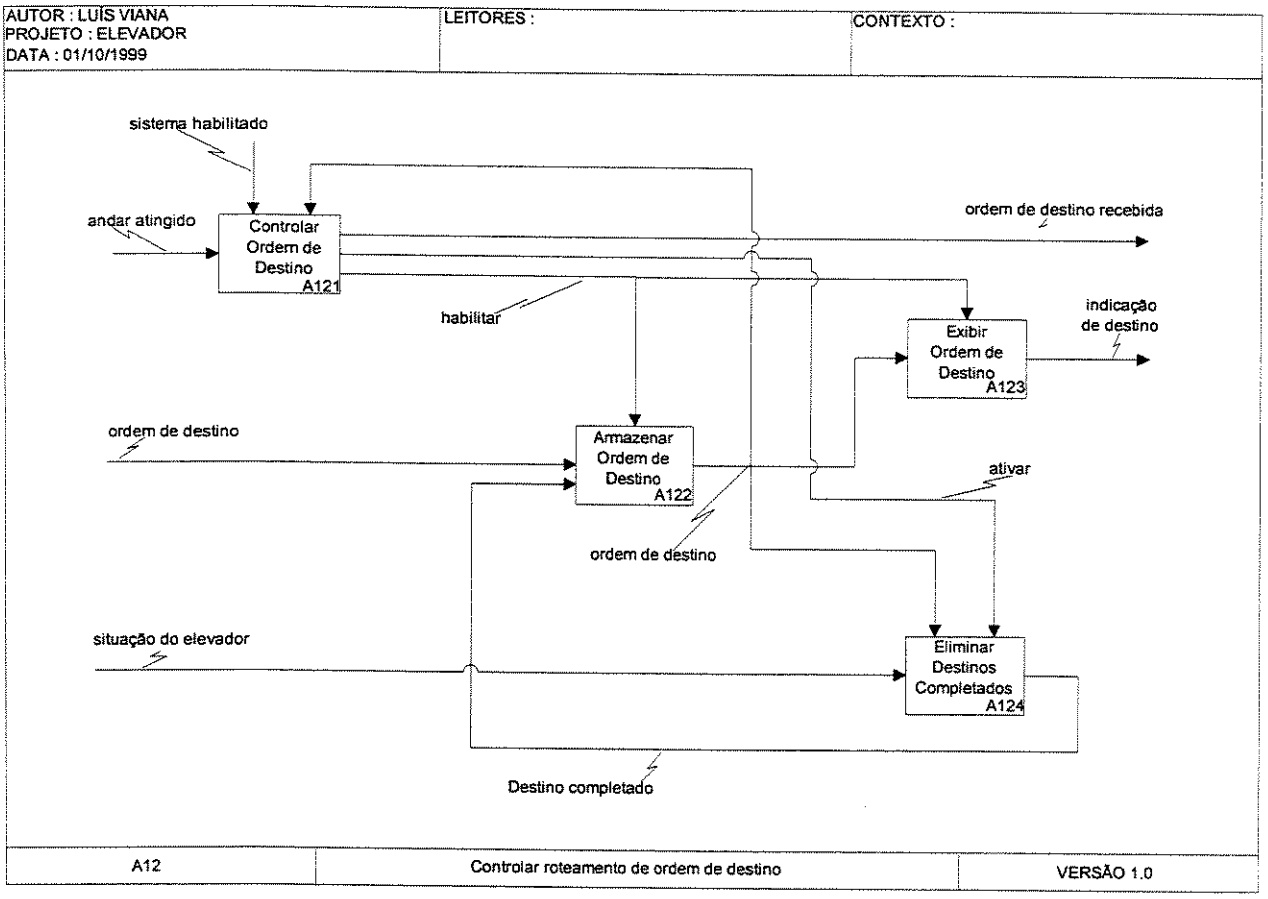


Figura 2. Actigrama SADT (adaptado de [YOU89]).

estado-transição (Figura 3) para descrever estados bem definidos de uma função de processamento (proveniente de um diagrama de fluxo de dados) e as condições de transição entre estes estados. Na verdade, é possível utilizar diversos paradigmas para descrever o comportamento dessas funções, de acordo com a natureza do sistema em questão [EDW93]. Particularmente, o GRAFCET – *Graphe Fonctionnel de Commande Etape-Transition* (Grafo Funcional de Comando Etapa-Transição) [IEC88], largamente empregado em sistemas automáticos seqüenciais, é bastante adequado para descrever a evolução de funções lógicas do comando de um SAP, sobretudo quando se trata de sistemas com estruturas de comando fortemente paralelas e/ou sincronizadas.

3) **Modelos informacionais.** Muitas vezes é preciso também definir a estrutura das relações entre os diferentes conjuntos de dados, visando o projeto futuro de bancos de dados. Isso é assegurado por diagramas do tipo entidade-relação [CHE76].

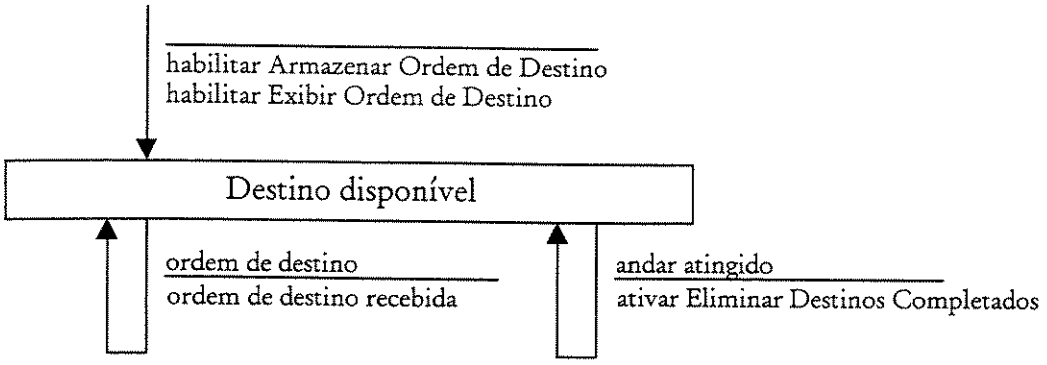


Figura 3. Diagrama estado-transição da função Controlar Ordem de Destinação (atividade A121 da Figura 2) [YOU89].

A arquitetura funcional, como um todo, é formada então por um conjunto coerente e complementar de modelos funcionais, comportamentais e, eventualmente, informacionais. A Figura 4 mostra de forma esquemática a arquitetura funcional de um SAP, descrita por um nível hierárquico de actigramas SADT e por um GRAFCET que modela o comportamento do tratamento T3 em relação a suas entradas e saídas.

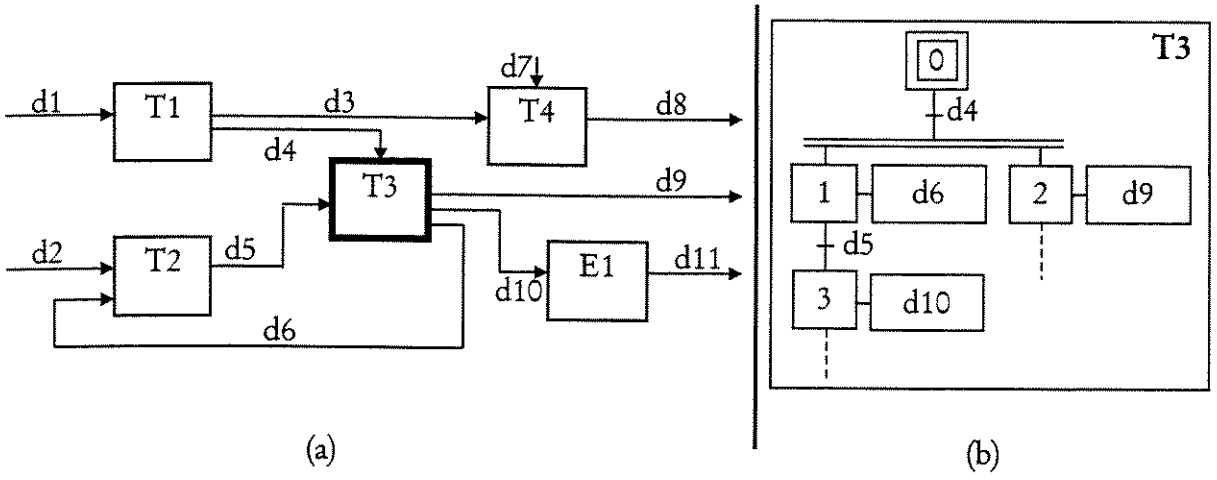


Figura 4. Modelo esquemático de arquitetura funcional: (a) fluxo de dados entre funções; (b) GRAFCET da função de tratamento T3.

2.2. Arquitetura Material

O que aqui se denomina **arquitetura material** (Figura 5) é a representação dos meios físicos disponíveis para resolver um problema específico em automação. Esses meios são

equipamentos de controle-comando, de informática industrial e de informática de gestão. Na verdade, é a experiência do projetista que lhe permitirá escolher entre fabricantes diversos ou componentes diferentes. As especificações do sistema ou mesmo restrições contratuais podem também impor toda ou parte da arquitetura material.

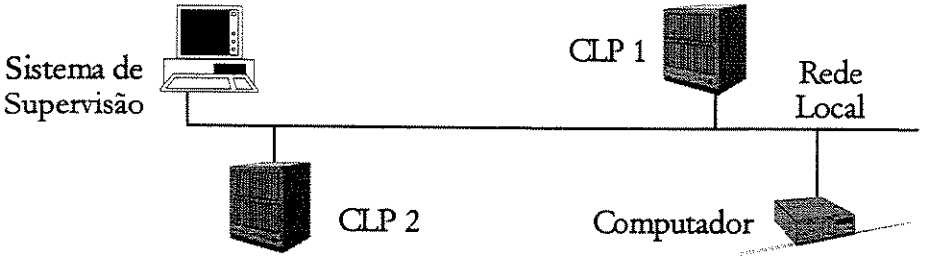


Figura 5. Exemplo de arquitetura material.

Apesar da diversidade dos equipamentos utilizáveis na parte de comando de um SAP, os diferentes componentes da arquitetura material podem ser agrupados em categorias genéricas:

- processadores (controladores lógicos programáveis (CLPs), computadores industriais diversos, etc.);
- interfaces homem-máquina (consoles de comando, terminais de diálogo, sistemas de supervisão, etc.)
- meios de comunicação (ligações ponto-a-ponto, redes, etc.)

2.3. Arquitetura Operacional

A partir do momento em que a arquitetura funcional está bem especificada e que uma primeira arquitetura material foi estabelecida, a tarefa da equipe de desenvolvimento consiste em projetar a arquitetura funcional sobre a arquitetura material, criando o que se chamará de **arquitetura operacional**. Esta constitui um modelo da repartição dos tratamentos e dos dados do sistema de comando nos equipamentos disponíveis pela associação das entidades da arquitetura funcional com os elementos da arquitetura material. A Figura 6 mostra um exemplo de arquitetura operacional proveniente das duas arquiteturas precedentes.

Para cada um dos tratamentos, determina-se qual processador deve assegurar sua execução. Cada armazenamento pode ser realizado ou por um equipamento de tratamento, por exemplo em sua memória, ou por um equipamento específico de armazenamento.

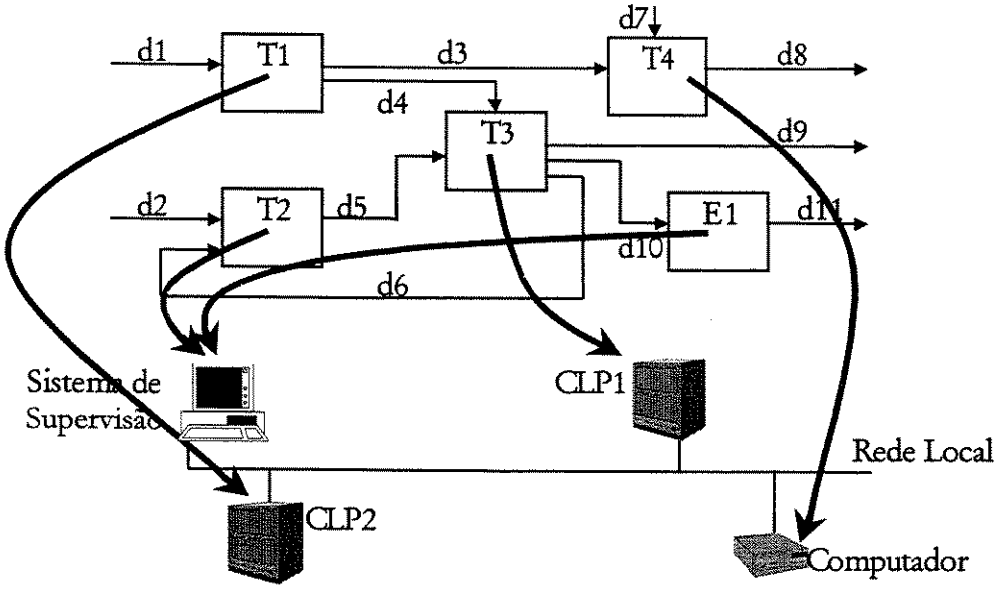


Figura 6. Exemplo de arquitetura operacional.

2.4. Verificação e Validação das Arquiteturas Operacionais

As atividades de verificação e de validação contribuem para a melhora da segurança de funcionamento dos sistemas automatizados por reforçarem a confiança, por parte dos utilizadores e projetistas, nos modelos estabelecidos para estes sistemas [ZAY97].

A *verificação* é a prova da coerência intrínseca de um modelo, independentemente do contexto de sua utilização. No caso de um modelo de arquitetura operacional, isso consistiria em, por exemplo:

- estabelecer uma decomposição clara das funções entre os diversos níveis hierárquicos dos diagramas de fluxo de dados, assim como assegurar a correspondência entre todas as entradas e saídas entre esses níveis;
- quanto aos modelos comportamentais, tratando-se de GRAFCETs, consistiria em

determinar suas propriedades de *reinicialização* (possibilidade de retorno ao estado inicial), *estabilidade* (a não existência de situações totalmente instáveis) e *bloqueio, total ou parcial* (existência de situações a partir das quais a evolução de todo o GRAFCET, ou de uma de suas partes, não mais é possível);

- assegurar o balanceamento dos deferentes tipos de modelos [YOU89] [EDW93] da arquitetura funcional, ou seja, verificar se todos os diagramas de fluxo de dados, diagramas estado-transição e diagramas entidade-relação são complementares entre si;
- verificar se os equipamentos da arquitetura material são fisicamente compatíveis;
- verificar se as funções dos equipamentos escolhidos são compatíveis com os processos de tratamento e de armazenamento especificados pela arquitetura funcional.

A *validação* de um modelo consiste em assegurar que ele é suficientemente preciso dentro de seu domínio de aplicação.

Por exemplo, a Figura 7 mostra o processo de reportagem de uma informação por um jornalista inglês e sua tradução para o francês por uma secretária bilíngüe [NEE87]. Aplicando os conceitos supracitados a este exemplo, o objetivo da verificação seria confirmar que o texto em francês (modelo de saída) constituísse uma tradução correta da fonte original em inglês. O processo de validação, por sua vez, asseguraria que a notícia nos jornais franceses seria uma descrição correta dos fatos que teriam ocorrido no mundo real.

Figura 7. Verificação versus validação [NEE87].

No projeto da parte de comando de um SAP, a validação é, então, a prova da adequação entre os modelos e os pré-requisitos do sistema que os primeiros devem representar. Um modelo de arquitetura operacional deve ser validado em relação aos pré-requisitos funcionais e em relação aos níveis de desempenho requeridos pela aplicação do SAP em questão.

2.5. Conclusão

A problemática deste trabalho é a avaliação de arquiteturas operacionais. O processo de avaliação é realizado, obviamente, sobre modelos já validados. No capítulo seguinte, serão apresentadas técnicas utilizáveis para validar modelos de arquitetura operacional, através da medida de indicadores de desempenho, e para avaliar e escolher uma arquitetura de comando apropriada ao SAP.

Capítulo 3

Técnicas para Avaliar o Desempenho de Arquiteturas de Comando

As técnicas mais usuais para a avaliação de desempenho de um Sistema Automatizado de Produção são:

- medidas por instrumentação;
- validação analítica;
- simulação.

Ao contrário de aprofundar-se nestas técnicas, o objetivo deste capítulo é apresentar de forma conceitual as principais diferenças entre estas abordagens e as implicações da escolha de uma delas.

3.1. Medidas por Instrumentação

A instrumentação é um dos meios que permitem medir o desempenho de um sistema automatizado e consiste, em parte, em utilizar sondas materiais ou, atualmente cada vez mais, sondas de *software*, sob a forma de microcódigo [FDI89]. Logo, a utilização desta técnica pressupõe que se construam protótipos físicos da arquitetura operacional.

As sondas materiais são geralmente sondas elétricas de alta impedância conectadas ao equipamento. Elas podem ser utilizadas para coletar dados sobre o estado dos componentes materiais do sistema (registros de memória, transmissão de dados, etc.). O termo *monitor material* faz referência a um sistema de medição que utiliza sondas materiais. Ele é externo ao sistema medido e não interfere com este, portanto não modifica seu desempenho. Como técnica

de validação, os monitores materiais geralmente não têm acesso às informações do software de comando, apresentam dificuldade de integração e custo não desprezível pois, para serem utilizados nas fases de projeto, seria preciso construir protótipos funcionais do sistema real e, em seguida, localizar pontos apropriados à medida de indicadores de desempenho.

As sondas de software são feitas por meio de instruções adicionadas ao sistema a ser medido para coletar dados de desempenho. Elas podem coletar dados através da leitura de registros de memória ou de informações de estado e enviar os indicadores correspondentes segundo as instruções contidas em seu código. Estas instruções ou tarefas suplementares a serem executadas (provenientes do "código espião") podem modificar o desempenho do sistema: o desempenho medido pode ser diferente do desempenho real. Um outro problema das sondas por microcódigo provém de rotinas defeituosas, que constituem uma contribuição não previsível à carga de CPU do sistema.

3.2. Validação Analítica

Os métodos analíticos são utilizados sobre modelos construídos completamente segundo uma representação matemática, por exemplo através de equações diferenciais ou fórmulas algébricas diversas.

Em razão da própria natureza da ferramenta matemática, as soluções fornecidas pela validação analítica são explícitas e exatas, e cobrem todo o espaço de pesquisa definido pelas hipóteses feitas durante a construção de um modelo. No entanto, a aptidão dos métodos analíticos para avaliar o comportamento de um determinado sistema depende intrinsecamente do grau de adequação entre estas hipóteses, das quais derivam as equações e fórmulas matemáticas, e o conjunto dos diferentes estados que podem surgir durante a evolução dinâmica do sistema em questão.

A crescente complexidade dos sistemas de informática atuais, como as redes de comunicação, torna muito difícil a construção de modelos matemáticos que descrevam o conjunto de estados possíveis da parte de comando de um SAP. Por exemplo, as classes de *redes de filas* que podem ser resolvidas analiticamente não são numerosas e não permitem que se levem em

consideração alguns comportamentos bastante difundidos em sistemas de informática [ROB94]. Pode-se citar a possessão simultânea de vários recursos , o bloqueio de clientes, a concorrência pelo acesso a recursos compartilhados (gestão de semáforos) ou os comportamentos de dependência entre clientes (sincronização, geração de clientes, entre outros). Ignorar esses comportamentos poderia conduzir a erros de avaliação do desempenho de um sistema.

3.3. Simulação de Sistemas

A simulação consiste em construir um modelo num computador e fazê-lo evoluir artificialmente segundo uma escala de tempo controlada [NEE87]. Em fase de projeto de sistemas, o processo de simulação é geralmente iterativo: constrói-se um modelo, este é simulado, coletam-se os dados provenientes deste ciclo de simulação, revisa-se o modelo e continua-se com as iterações até uma adequação razoável entre o modelo e o sistema estudado.

Apesar de não ser a simulação um método exato, como a validação analítica, nem direto, como as medidas por instrumentação, muitas vezes simular o comportamento de um sistema é a única técnica aplicável.

Ao contrário das medidas por instrumentação, a simulação permite que se estude um sistema real sem que seu comportamento seja perturbado. Além disso, nos casos em que um estudo direto do fenômeno modelado poderia ter conseqüências graves no mundo real, ou nos casos em que a construção de protótipos não é viável, a simulação permite que se prevejam situações excepcionais que não poderiam ser reproduzidas na realidade sem riscos ou custos exagerados ou inaceitáveis.

A técnica de validação analítica, por mais poderosa que seja, não se presta a modelar todos os comportamentos dos sistemas de informática pelas razões mencionadas anteriormente. E mesmo em casos de sistemas passíveis de serem modelados por relações matemáticas, as soluções muitas vezes requereriam páginas de cálculo ou uma capacidade muito grande de memória. Nesses casos, a simulação poderia fornecer soluções razoáveis, mesmo que aproximadas, sem tais custos de tempo ou de recursos.

Considere-se o caso de uma rede de informática [TOU91]. Deseja-se conhecer a máxima taxa de transferência de um controlador de acesso. A construção de um protótipo do controlador de acesso e os recursos apropriados para analisar a rede permitiria que se observassem as características do sistema estudado (por exemplo, o número de pacotes de dados que circulam na rede). Este método, no entanto, requer um tempo relativamente longo para ser posto em prática, é custoso e pouco flexível. Ele não permite também que se obtenham informações sobre o comportamento interno do controlador de acesso (por exemplo, qual o número de *buffers* utilizados, quantas cópias de mensagens foram feitas ou quanto tempo uma mensagem espera numa fila). É possível conectar sondas à placa principal e, através da medidas de tensão elétrica, depois de um tratamento adequado, compreender o que se passa na placa. Isto requereria meios físicos de complexidade não desprezível e gerarim a problemática da coleta e do tratamento de resultados. Seria possível também modificar o código executado na placa e adicionar funções de rastreio e contadores. Mas o acréscimo das instruções de controle ao código do protocolo provavelmente deixaria mais lenta sua execução, o que poderia resultar em modificações de seu comportamento.

Uma melhor solução, neste caso, consistiria em modelar o comportamento do ambiente e do controlador de acesso em computador para que se pudesse agir sobre todos os parâmetros deste universo recriado. Como a evolução do tempo no modelo seria controlada, a coleta de indicadores de desempenho poderia ser realizada num tempo de simulação bastante reduzido sem perturbar o comportamento do protocolo.

No entanto, não se deve esquecer que a simulação não fornece resultados gerais para um determinado problema. Como a cada execução o computador faz com que o modelo evolua diferentemente de um estado a outro, entre os diferentes estados acessíveis, um grande número de ciclos de simulação é necessário para obter resultados com um intervalo de confiança suficientemente restrito [PRI86].

Ao contrário da validação analítica, nem a simulação nem a técnica de medidas por instrumentação são capazes de abranger todos os estados possíveis de um sistema real: a simulação, devido à sua natureza probabilística; as medidas, devido ao caráter pontual dos dados coletados. Encontra-se em [TOU91] uma figura que representa de forma simplificada as

diferenças entre a validação analítica e a simulação, e à qual adicionou-se, no presente trabalho, uma representação dos estados passíveis de serem avaliados através de medidas diretas. Logo, a Figura 8 representa de maneira esquemática o espaço dos estados possíveis do sistema e os diferentes métodos de avaliação de desempenho:

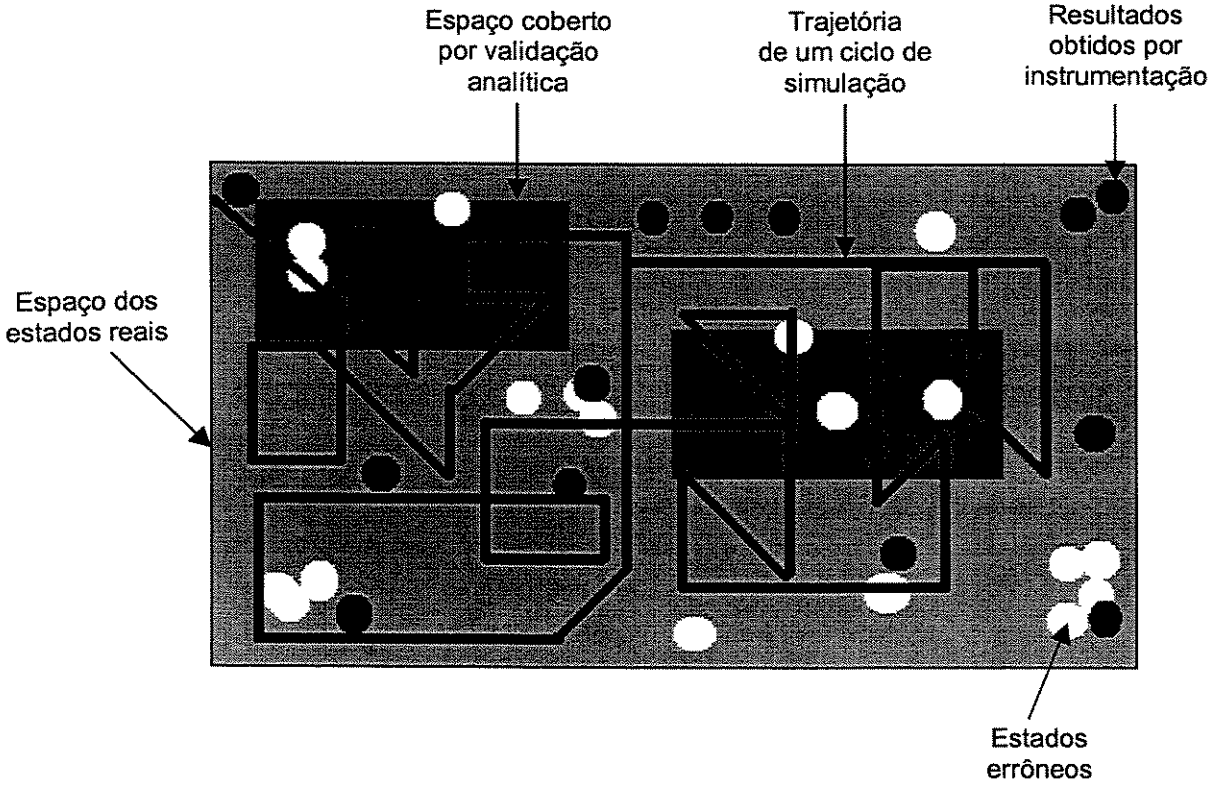


Figure 8. Representação esquemática dos métodos de validação.

- os retângulos hachurados representam os estados do sistema que podem ser estudados através de técnicas de validação analítica; cada conjunto de hipóteses cobre completamente um espaço de pesquisa determinado;
- os pontos negros representam as medidas por instrumentação, material ou por *software*; as informações referentes aos diferentes estados são atingidos segundo as diferentes condições de medição e as condições de entrada impostas ao sistema;
- o "caminho" em traço negro representa os estados do sistema atingidos por técnicas de simulação; mais precisamente, cada ciclo de simulação "percorre" uma sucessão de estados acessíveis;
- os pontos brancos representam os estados em que o sistema apresenta erros.

3.4. Conclusão

O processo de projeto proposto neste trabalho situa-se no começo do ciclo de vida de um SAP. Como as medidas por instrumentação, assim como a validação analítica requerem grandes investimentos da parte da equipe de projeto, a simulação torna-se a técnica mais adequada para avaliar as opções possíveis de arquitetura operacional de forma flexível e com custos relativamente baixos. Os próximos capítulos tratam de diferentes paradigmas de simulação.

Capítulo 4

Paradigmas de Simulação

A utilização de diferentes paradigmas de simulação depende principalmente do problema específico que se deseja estudar. Particularmente, a simulação de sistemas de manufatura pode ser dividida em três classes complementares:

- a simulação de sistemas automáticos, utilizada para estudar a lógica do controle dos procedimentos de produção;
- a simulação de fluxo, que permite o estudo do sistema de produção como um todo e mais precisamente os aspectos funcionais ligados ao fluxo de produtos;
- a simulação em robótica através de CAD, que considera a cinemática dos meios produtivos e fornece uma visualização de sua evolução no espaço.

A seguir serão apresentados dois dos paradigmas mais utilizados na simulação de sistemas industriais: as *redes de filas* e as *Redes de Petri*.

As primeiras foram primeiramente utilizadas na modelagem e na simulação em manufatura e tiveram sua área de aplicação estendida à modelagem de sistemas de informática pela analogia existente entre, por exemplo, os fluxos de produtos numa fábrica segundo uma determinada política de ordenamento e o roteamento de pacotes de informação segundo um determinado protocolo de comunicação.

Primeiramente utilizadas em modelagem qualitativa (por exemplo, verificação de

protocolos, etc.) as Redes de Petri (RdP) são cada vez mais aplicadas à análise quantitativa de sistemas de informática, depois que extensões em seu comportamento original, desenvolvido na década de sessenta [BRA83], foram introduzidas visando considerar o aspecto de tempo (RdP temporizadas) e comportamentos estocásticos (RdP estocásticas). As RdP são particularmente aptas à descrição de fenômenos de concorrência, conflito e sincronização.

4.1. Redes de Filas

Num sistema de informática, as redes de filas permitem a representação da circulação de dados. Elas possibilitam que se obtenham resultados quantitativos sobre os sistemas modelados. Tipicamente, a modelagem através de redes de filas fornece informações do tipo:

- qual o número médio de clientes à espera de um recurso como uma CPU;
- que efeitos sobre os tempos de resposta do sistema terá a adição de um recurso suplementar, como um segundo disco; etc.

Os modelos em forma de redes de filas são compostos de dois tipos de entidades [FDI89]: os *clientes* e as *estações de trabalho*. Os clientes são as entidades passivas que modelam a informação circulando dentro do sistema. As estações de trabalho são decompostas em duas partes: os *servidores* e as *filas*. Os servidores são as entidades ativas que modelam a maneira de produzir, de processar ou de modificar a informação veiculada pelos clientes. As entidades *filas* possibilitam armazenar os clientes que esperam por processamento. Como os servidores podem processar simultaneamente apenas um número limitado de clientes, se a taxa de chegada de clientes é superior à taxa com que os servidores processam os clientes (ou seja, a taxa com que os clientes deixam a estação), as ordens de processamento acabarão por se acumular a montante do servidor.

Tome-se como exemplo o caso de um comutador de rede de comunicação cuja função principal é o roteamento de pacotes de informação que recebe, para as linhas de saída.

A Figura 9 mostra uma primeira abordagem de modelagem a de um sistema assim por meio de uma fila simples. Os pacotes chegam aleatoriamente segundo uma média λ pacotes por

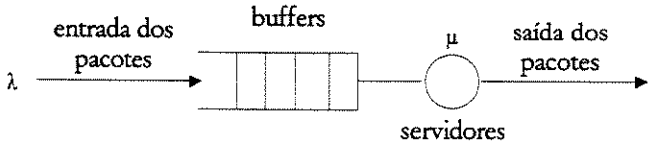


Figure 9. Modelagem de um comutador de rede de comunicação [FDI89].

unidade de tempo. Elas em seguida esperam para serem processadas, ou seja, serem transmitidas para a linha de saída que lhes permitirá atingir o destinatário; para isso, elas são memorizadas nos buffers do comutador. Elas serão em seguida processadas segundo uma determinada ordem de serviço, com um tempo médio igual a $1/\mu$ unidades de tempo. Isso significa uma capacidade média de processamento de μ pacotes por unidade de tempo.

Uma rede de filas é caracterizada por um **trajeto seguido pelos clientes** no sistema, por diferentes **classes de clientes** e pelo **comportamento das estações**.

O trajeto seguido pelos clientes pode ser estático (o cliente vai de estação em estação segundo uma rota predeterminada) ou dinâmico (depois do processamento em uma estação, o cliente tem a escolha entre diversas outras estações para continuar seu processamento). Esta escolha é geralmente ponderada por funções probabilísticas. Diversas classes de clientes podem estar presentes no sistema. Todos os clientes pertencendo a uma mesma classe seguirão o mesmo trajeto (definido de forma estática ou dinâmica).

A descrição do comportamento de uma estação compreende diversos parâmetros. A notação de Kendall [ROB94] distingue os seis parâmetros seguintes:

- a lei de chegada de clientes numa estação (exponencial, constante, etc.);
- o tempo de serviço dos clientes num servidor, segundo o mesmo conjunto de leis de chegada de clientes;
- a capacidade dos servidores; por exemplo, o número máximo de clientes que podem ser processados simultaneamente;

- a capacidade máxima das filas (os clientes suplementares chegando quando a fila está cheia podem ser perdidos ou encaminhados para uma outra estação determinada, dependendo dos recursos do modelo de simulação);
- o número total de clientes no sistema;
- a disciplina de serviço, cujas principais modalidades são:
  - FIFO (*First In, First Out*),
  - LIFO (*Last In, First Out*),
  - PS (*Processor Sharing*), duração proporcional ao número de clientes presentes na estação,
  - por prioridades (com ou sem preempção, ou seja, precedência predeterminada), em que o nível de prioridade depende da classe à qual pertence o cliente.

As redes de filas são representadas graficamente por um conjunto de estações (Figura 10). As flechas representam o trajeto seguido pelos clientes. Os números associados às flechas indicam a probabilidade de escolha da estação seguinte. Na figura, os clientes chegando ao sistema são processados pelo servidor 1. Eles se dirigem em seguida para uma dentre as estações 2, 3 e 4. As probabilidades de escolha das estações 2, 3 e 4 são, respectivamente, 50%, 30% e 20%. Os clientes são em seguida processados todos pelo servidor 5 e depois deixam o sistema segundo uma probabilidade de 80% ou recomeçam todo o processo passando pelo servidor 6.

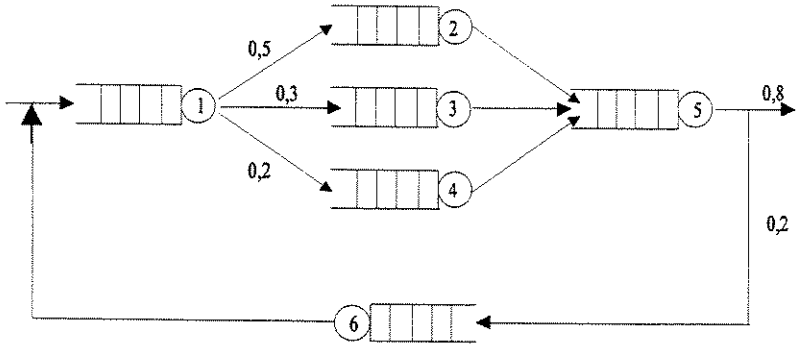


Figura 10. Exemplo de rede de filas.

4.2. Redes de Petri

O conceito das Redes de Petri é originário de dissertação de Carl Adam Petri, apresentada em 1962 perante a Universidade Técnica de Darmstadt, na época Alemanha Ocidental, versando seus trabalhos como cientista na Universidade de Bonn [BRA83].

As Redes de Petri (RdP) constituem um paradigma gráfico e matemático aplicável a diversos tipos de sistemas: concorrentes, assíncronos, paralelos, determinísticos ou estocásticos.

Particularmente, as RdP são bem adaptadas à expressão do paralelismo e da sincronização presentes em sistemas de informática. Elas são empregadas, por exemplo, na avaliação de desempenho de arquiteturas distribuídas, na modelagem de protocolos de comunicação, etc.

A modelagem por meio de RdP pode auxiliar na resposta de questões como [REI82]:

- se o sistema modelado resulta em bloqueios (neste caso, a RdP não apresenta totalmente a propriedade de **"liveness"**); isto pode significar que um protocolo não é confiável e que ele pode conduzir a panes;
- se o número de entidades cresce indefinidamente (a RdP não é **limitada**); isso pode significar que o sistema adquire recursos que não são liberados;
- se algumas partes do modelo são realmente acessíveis durante a vida do sistema (a propriedade de **alcançabilidade** da RdP);
- se o sistema termina sua execução retornando a seu estado inicial (se a RdP é **reversível**);
- quais as condições em que o sistema pode aceitar uma nova requisição;
- se o sistema passa por ciclos durante sua evolução; etc.

Segundo seu conceito original, as RdP são constituídas de quatro tipos de entidades: os *lugares*, as *transições*, os *arcos* e as *marcas*. Graficamente, os lugares são representados por círculos e as transições por traços ou retângulos. Os arcos são orientados e ligam os lugares às transições, e vice-versa. Este conjunto forma um gráfico bipartido: cada lugar só é ligado a transições e cada transição só é ligada a lugares. Os lugares contém um número inteiro (positivo

ou nulo) de marcas. Cada transição é ligada a uma dada quantidade de lugares ditos *de entrada* e *de saída* representando, respectivamente, as *pré-condições* e as *pós-condições* deste subsistema. Cada arco, chamado então de *arco de entrada* ou *arco de saída*, tem associado a si um determinado número de marcas, denominado seu *peso*.

A regra fundamental da teoria das RdP é a do disparo das transições. Uma transição é dita *habilitada* se cada um dos lugares a montante desta transição contém uma quantidade de marcas igual ou superior ao peso do arco de entrada correspondente.

O disparo de uma transição consiste em retirar de cada lugar a montante desta um número de marcas igual ao peso do arco de entrada correspondente, e em adicionar a cada lugar a jusante um número de marcas igual ao arco de saída correspondente.

A Figura 11 mostra as condições para o disparo de uma transição. (a) e (b) mostram estados em que a transição não é *habilitada*. O estado representado por (c) permite seu disparo e, finalmente, (d) representa o estado da RdP depois do disparo da transição.

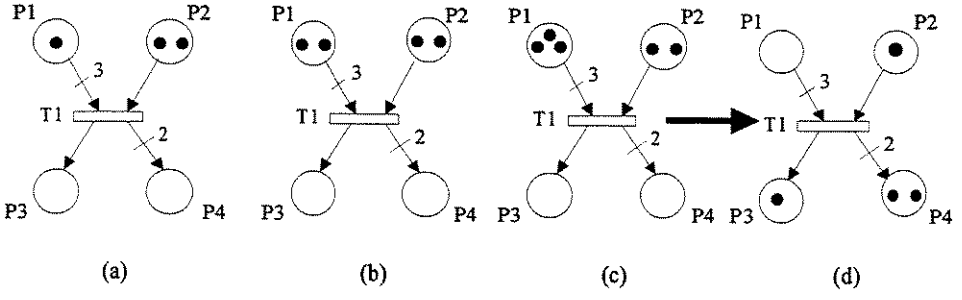


Figura 11. (a) e (b) transição não habilitada; (c) transição habilitada; (d) RdP depois do disparo.

As RdP, do ponto de vista matemático, possuem propriedades que se dividem em dois grupos: propriedades dependentes da marcação (propriedades comportamentais) e as não dependentes da marcação (propriedades estruturais). As propriedades mencionadas no começo deste subcapítulo (*"liveness"*, *limitação*, *alcançabilidade* e *reversibilidade*), entre outras, pertencem ao primeiro grupo. Entre as propriedades estruturais, por sua vez, podem ser citadas *limitação estrutural*, *conservação*, *repetitividade* e *persistência* [BRA83].

Essas propriedades das RdP são extremamente úteis para analisar os sistemas modelados. Um importante aspecto a ser considerado durante análise é se existe correspondência biunívoca entre o modelo sob forma de RdP e os pré-requisitos funcionais da aplicação em questão. Este é um ponto crítico, principalmente quando modelos de grande extensão são construídos para sistemas complexos.

A simulação por eventos discretos é o método utilizado neste trabalho para avaliar os modelos dinâmicos sob forma de RdP. No entanto, existem outros métodos de análise das propriedades das RdP. Entre eles, os principais são:

- árvore de cobertura;
- análise de invariantes;
- redução de redes.

O método de **árvore de cobertura** consiste, basicamente, em enumerar todas as possíveis marcações passíveis de serem atingidas a partir da marcação inicial de uma RdP. Começando da marcação inicial, procura-se construir um conjunto de cobertura através do disparo de todas as transições sucessivamente habilitadas.

A **análise de invariantes** parte do princípio de que os arcos descrevem relações entre os lugares e as transições, e podem ser representados por duas matrizes. Estudando as matrizes e as equações lineares baseadas na regra de execução das RdP, pode-se encontrar subconjuntos de lugares nos quais a soma de marcas permanece inalterada (conceito de invariante de lugar). De forma análoga, pode-se encontrar uma seqüência de disparos de transições que faz uma marcação voltar a um mesmo valor (conceito de invariante de transição).

A abordagem de **redução de redes** consiste em simplificar a estrutura de uma rede complexa preservando suas propriedades e, através da rede reduzida, derivar as propriedades da rede original.

Finalmente, a **simulação**, como método de análise, faz a RdP evoluir através da execução de seu algoritmo. Como mencionado no capítulo anterior, esta não é uma método adaptado para

provar a correção do modelo em casos gerais (ponto forte de métodos baseados da técnica de validação analítica, como a análise de invariantes), mas permite derivar o desempenho de um sistema sob premissas bastante realistas.

4.3. Conclusão

As redes de filas são utilizadas para modelar redes de comunicação, mas sua estrutura não dispõe da flexibilidade que têm as RdP. Apesar desta flexibilidade, os modelos baseados em RdP apresentam a tendência a serem muito grandes mesmo para sistemas de tamanho modesto. A ausência do aspecto temporal no conceito original das RdP, por sua vez, limita-as à modelagem meramente qualitativa de sistemas. Medidas de desempenho baseadas em tempo de operação, entre outras, tornam-se impraticáveis. Visando resolver estas limitações, algumas extensões foram desenvolvidas, a saber as RdP coloridas e as RdP temporizadas, tratadas no próximo capítulo.

Capítulo 5

Redes de Petri coloridas e Temporizadas

Neste capítulo, são apresentadas duas extensões das Redes de Petri (RdP) que possibilitam a construção e a avaliação de modelos de sistemas dinâmicos complexos: a coloração de marcas e a temporização associada às entidades da RdP.

Para melhor demonstrar o poder destas extensões, elas serão ilustradas por meio de dois exemplos sobre compartilhamento de diferentes recursos por diferentes tipos de processos, estes últimos com durações variáveis.

Estes modelos foram construídos por meio do programa ***Design/CPN*** [DES93], em que CPN significa *Colored Petri Nets*. Design/CPN é um programa dedicado à modelagem e simulação de RdP coloridas e temporizadas, desenvolvido originalmente pela empresa *Meta Software Corporation* (Cambridge, MA, E.U.A.) e cujo desenvolvimento atual está aos encargos do grupo CPN da Universidade de Aarhus, Dinamarca. As RdP construídas com Design/CPN são expressas através da linguagem CPN/ML, uma extensão da linguagem de programação ML (*Meta Language*) com modificações próprias a Design/CPN; e é a sintaxe desta linguagem que estará presente nos exemplos a seguir.

5.1. Redes de Petri Coloridas

Durante a modelagem de um sistema real, freqüentemente o tamanho da RdP se torna muito grande. Este fenômeno de crescimento, na maioria dos casos, nasce da repetição de sub-

redes de estrutura idêntica que servem unicamente a indicar estados diferentes do sistema. As RdP coloridas são uma das abreviações do conceito original que possibilitam, sem mudar o poder do algoritmo deste conceito, nem suas propriedades, uma economia de escrita e de leitura extremamente necessária à representação de sistemas complexos.

As RdP coloridas seguem o seguinte princípio: a marcação de uma rede representa em geral o número de acontecimentos de eventos de um mesmo tipo. Associando a cada tipo de evento um atributo distintivo (que por razões históricas foi chamado de *cor*), a marcação de um único lugar pode então modelar diversos acontecimentos de eventos de tipos diferentes, através de marcas de cores diferentes (na verdade, segundo a definição das RdP coloridas, um lugar deve ter a cor das marcas que pode conter definida a priori [JEN87]; este lugar poderá conter, no entanto, todas as subcores que tenham sido definidas como "filhas" desta "cor mãe").

Para as RdP coloridas, uma transição é *habilitada* não somente se os lugares a montante possuem o número de marcas exigidas pelos arcos de entrada, como no conceito original, mas ainda se estes lugares contêm marcas de tipos, ou seja, portando as *cores*, exigidas pelos arcos de entrada.

Um exemplo da utilização de cores é mostrado na seção seguinte.

5.2. Exemplo de Aplicação de Redes de Petri Coloridas: Alocação de Recursos

O problema. Assume-se que existe um conjunto de processos que compartilham recursos comuns. Há dois tipos diferentes de processos (chamados de processos **p** e **q**) e três tipos diferentes de recursos (chamados de recursos **r**, **s** e **t**). Os processos poderiam representar diferentes programas de computador (por exemplo, editores de texto ou programas de CAD) enquanto os recursos poderiam ser diferentes meios físicos compartilhados por estes programas (por exemplo, discos, impressoras, etc.). Cada processo é cíclico e durante cada uma das partes de seu ciclo, ele precisa obter acesso exclusivo a uma quantidade variável de recursos.

A estrutura do modelo. O sistema de alocação de recursos é modelado pela RdP colorida da Figura 12. Os processos podem estar em cinco estados diferentes, representados pelos lugares

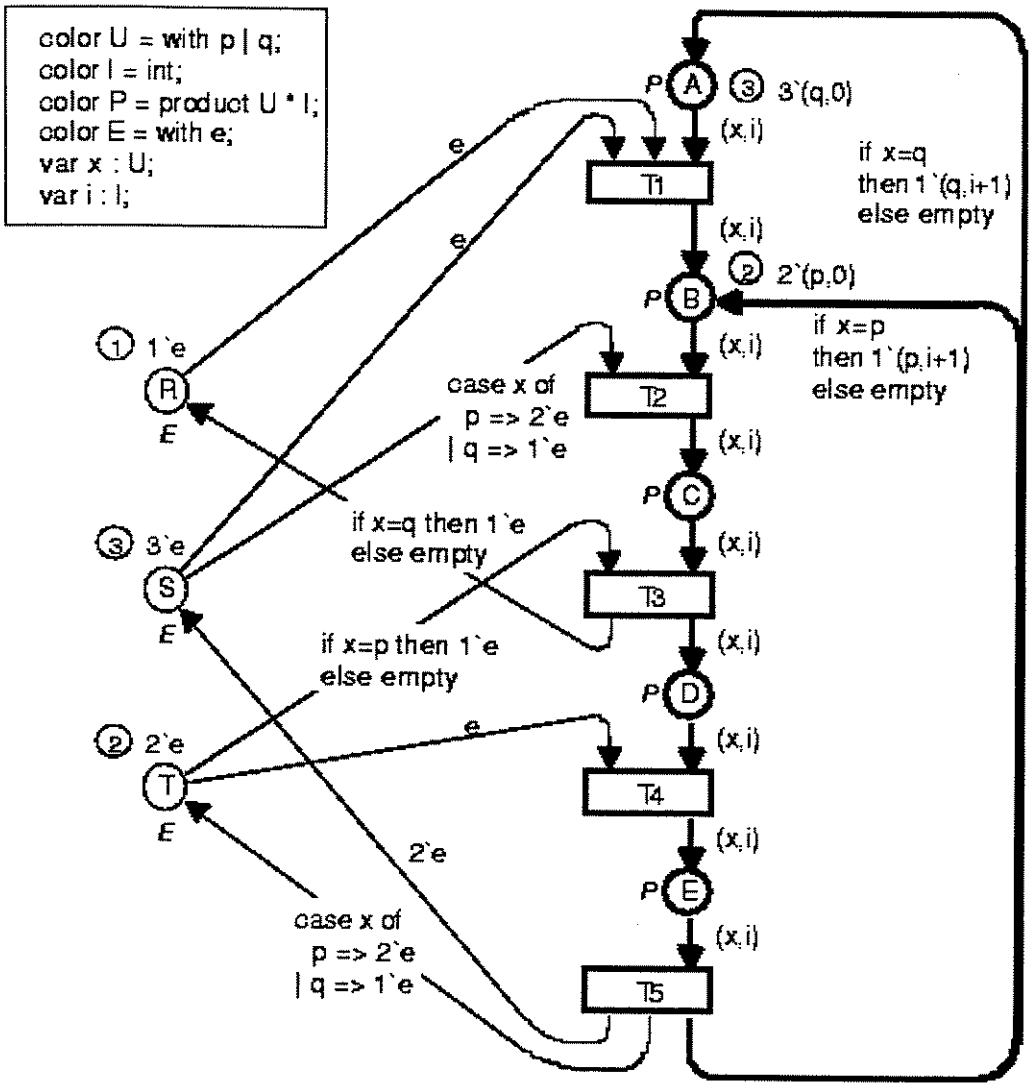


Figura 12. Exemplo de RdP coloridas: um sistema com alocação de recursos.

A-E. As marcas em um lugar são representadas por um pequeno círculo contendo um número (indicando um número inteiro de marcas) e uma parte de texto (indicando as **cores individuais destas marcas**).

Como explicado anteriormente, todas as marcas em um determinado lugar devem ter a cor de um tipo específico (devem ter subcores desta cor). Esta cor é chamada de **cor do lugar**. Na figura, estas cores estão escritas em itálico: os lugares A-E (modelando os diferentes estados dos processos) têm cor *P*, e os lugares R-T (modelando os diferentes recursos requisitados pelos

processos) têm cor *E*.

As declarações do conjunto de cores (no quadro na parte superior, à esquerda) indicam que cada marca em A-E tem uma cor que é uma dupla (pois a cor *P* é declarada como o produto cartesiano das cores *U* e *I*). O primeiro elemento da dupla é um elemento de *U*, portanto este é ou **p** ou **q**, e o segundo elemento indica o número total de ciclos que o processo completou. Nota-se também que todas as marcas nos lugares R-T têm a mesma cor (*e* é o único elemento de *E*). Isto quer dizer que estas marcas não portam nenhuma informação, a não ser sua presença ou ausência no interior de um do lugares R, S ou T (o que representa o número de recursos **r**, **s** ou **t** disponíveis).

A situação representada na figura constitui a marcação inicial do modelo. Há três marcas **(q,0)** em A e duas marcas **(p,0)** em B, enquanto C, D e E não possuem marcas. Por outro lado, R possui uma marca *e*, S possui três marcas *e* e T possui duas marcas *e* (representando a disponibilidade de um recurso **r**, três recursos **s** e dois recursos **t**, respectivamente).

Cada uma das cindo transições T1-T5 representa a evolução dos processos de um estado para o estado seguinte. As legendas sobre os arcos de uma transição indicam como os recursos são reservados ou liberados por um processo.

A dinâmica. Por exemplo, considere-se a transição T2, com dois arcos de entrada e um de saída. A legenda "(x,i)"aparece duas vezes (sobre o arco que parte de B e sobre o que entra em C). Estas legendas têm duas variáveis, *x* e *i*, e através das declarações é possível constatar que x é de tipo U enquanto i é de tipo I. O disparo de T2 transfere uma marca de B para C sem mudar sua cor, pois as duas legendas são idênticas. Por outro lado, segundo a legenda "case x of p =>2`e | q =>1`e" (sobre o arco que sai de S), o número de marcas retiradas do lugar S depende do processo em questão: um processo **p** precisa de dois recursos **s** (logo, retira de S duas marcas *e*) para passar de B para C, enquanto um processo **q** precisa de apenas um recurso **s** para passar de B a C (retirando, assim, uma marca *e* do lugar S).

Considere-se desta vez a transição T5. Esta transição transfere uma marca do lugar E ou para A, ou para B (processos **p** vão para B, enquanto processos **q** vão para A). Ao mesmo tempo,

T5 atualiza o contador de ciclos *i*. Além disso, nota-se que o disparo de uma transição pode gerar não apenas marcas de cores diferentes mas também quantidades diferentes de marcas. Por exemplo, o conjunto de marcas adicionadas/retiradas pelo disparo de uma transição (como T5) pode ser até vazio (*empty*).

Esta diferenciação de comportamento segundo as cores das marcas torna as RdP coloridas capazes de modelar diferentes subsistemas, como processos diversos, cuja evolução segue os mesmos algoritmos.

5.3. Redes de Petri Temporizadas

O conceito de tempo não é abordado na definição original das RdP. No entanto, para a avaliação de desempenho de sistemas, torna-se necessário introduzir intervalos temporais às transições e/ou lugares dos modelos. As RdP temporizadas possibilitam avaliar quantitativamente o desempenho temporal dos sistemas estudados.

As RdP temporizadas são redes não-autônomas [BRA83], ou seja, elas pertencem à classe de extensões das RdP cuja evolução dinâmica não depende unicamente do estado da rede em questão mas também do ambiente associado a ela. No caso específico das RdP temporizadas, este ambiente é caracterizado por um relógio global que determina a disponibilidade das marcas depois do disparo das transições.

Estas RdP são chamadas de *temporizadas* (determinísticas) se os intervalos são especificados de forma determinística ou *estocásticas* se estes intervalos são especificados de forma probabilística.

Nas RdP temporizadas, as regras de disparo de transições foram modificadas para levar em consideração a duração das atividades, ações ou estados de um modelo. Assim, se uma marca chega a um lugar num instante **t** e a transição TI (cujo disparo gerou esta marca) especifica uma atividade de duração $\mathbf{z}_{,,}$ esta marca somente estará "pronta" para disparar a transição TII seguinte no instante **t+z**. Isto porque as marcas em RdP temporizadas apresentam dois estados: disponível ou indisponível. A passagem do estado disponível ao estado indisponível acontece quando uma

transição dispara, ou seja, as marcas colocadas pela transição em um lugar de saída estão em estado indisponível. O estado indisponível corresponde à situação em que o disparo de uma transição representa uma atividade sendo executada. Considerando novamente as transições TI e TII, as marcas em um lugar passam do estado indisponível ao estado disponível assim que a ação associada a TI é finalizada. Então, elas podem ser utilizadas para disparar TII exatamente como nas RdP autônomas.

Fundamentalmente, a evolução das RdP temporizadas depende do **relógio global**. Os valores deste relógio representam o tempo no modelo. Além de portar cores, as marcas podem portar um valor de tempo, chamado também de selo de tempo. O selo de tempo de uma marca indica o mínimo valor de tempo do modelo em que a marca pode ser utilizada, ou seja, retirada do lugar em que ela está para disparar uma transição.

Assim, em uma RdP colorida e temporizada, uma transição é *habilitada* se:

- as marcas apresentam-se portando as cores e em número exigidos por cada arco de entrada;
- as marcas estão disponíveis, ou seja, **os selos de tempo das marcas a serem retiradas devem ter valor menor ou igual ao tempo do modelo.**

A modelagem de uma atividade/operação de Δr unidades de tempo é feita por meio de uma transição T que gera para os lugares de saída marcas cujos selos de tempo são Δr unidades de tempo superiores ao valor do relógio em que T é disparada. Como resultado do disparo, as marcas geradas por T serão indisponíveis durante Δr unidades de tempo.

A execução de RdP coloridas e temporizadas é similar às filas de eventos encontradas em várias linguagens de programação destinadas à simulação a eventos discretos. O modelo permanece num instante enquanto houver transições a disparar. Quando não há mais, o relógio é avançado até o próximo instante em que haverá transições disparáveis (logo, em que haverá marcas disponíveis).

Na seção seguinte é apresentado um exemplo proveniente do anterior, ao qual foi

adicionado o conceito de temporização.

5.4. Exemplo de Redes de Petri Coloridas e Temporizadas: Alocação de Recursos Temporizada

O problema. É o mesmo do exemplo da seção 5.2. Mas desta vez os estados do modelo têm durações explícitas e variáveis segundo o tipo de processo, **p** ou **q**. Conseqüentemente, a utilização dos recursos **r**, **s** e **t** tem durações distintas segundo cada um dos dois processos.

A estrutura do modelo. Neste sistema, modelado pela Figura 13, utiliza-se um relógio global que começa em 0. Através da terceira e quarta linhas das declarações, nota-se que as marcas da cor *P* são temporizadas (portam selos de tempo) ao contrário das marcas da cor *E*. Isto implica que as marcas de *E* estão sempre "prontas" para utilização. O pequeno quadro abaixo do quadro de declarações indica o tempo atual do modelo, que é 653. Os valores de tempo associados a cada marca são os números depois dos símbolos @ (*at*). Após o disparo de uma transição, o novo selo de tempo de uma marca é calculado pela soma entre o tempo atual do modelo e o intervalo indicado pela legenda sobre o arco de saída (depois do operador @+).

A dinâmica. Ao analisar, por exemplo, o lugar B, vê-se que ele contém três marcas – **(p,5)**, **(p,6)** e **(q,1)**. Somente (p,6) et (q,1) estão disponíveis neste momento (653), enquanto (p,5) porta um selo de tempo 658. Além disso, existe uma situação de conflito entre (p,6) e (q,1) porque T2 exige combinações de ambas as marcas com marcas de S para ser disparada. como resultado deste conflito, T2 será disparada retirando de B ou (p,6) sozinha, ou (q,1) sozinha (se houvesse três marcas em S, (p,6) e (q,1) seriam ambas consumidas ao disparar T2).

Os intervalos de disparo podem depender das cores das marcas de entrada ou de saída. Isto é ilustrado pelo arco de saída de T3, em que o intervalo depende do valor da variável *x*. Para processos p, o intervalo é de 13 unidades de tempo, enquanto para processos q, ele é igual a 9 unidades. Os intervalos são descritos por meio de expressões algébricas, e além disso Design/CPN permite a utilização de funções matemáticas e estatísticas.

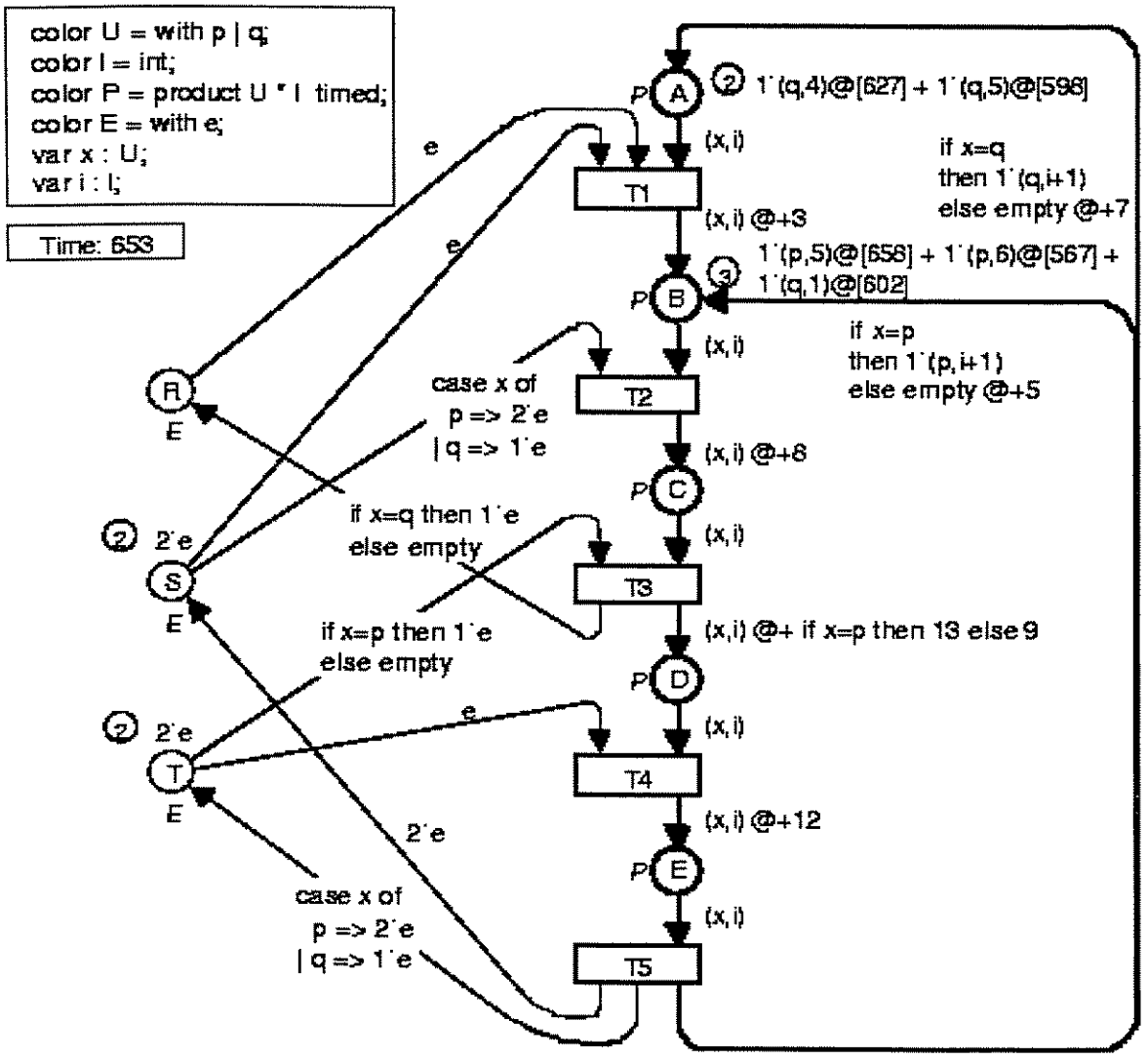


Figura 13. Exemplo de RdP coloridas e temporizadas.

5.5. Conclusão

Combinadas, a coloração de marcas (que torna um modelo compacto e genérico) e a temporização (tornando possível considerar a duração de atividades) fazem das RdP coloridas e temporizadas um paradigma poderoso para modelar sistemas dinâmicos complexos. Além disso, ferramentas computacionais como Design/CPN permitem controlar completamente a evolução de um modelo, resultando numa melhor coleta de dados de simulação.

Capítulo 6

Metodologia para Avaliação de Desempenho de Arquiteturas de Comando

Diversas configurações de arquitetura operacional podem surgir durante o período em que a equipe de projeto se propõe a fazer a repartição das funções de comando nos diferentes equipamentos disponíveis. Antes de passar às fases de construção da arquitetura de comando, é necessário validar essas configurações e em seguida avaliar os modelos de arquitetura operacional possíveis segundo critérios indicados pelas especificações do SAP, como o desempenho temporal do sistema.

6.1. Modelagem Dinâmica das Arquiteturas Operacionais

Para avaliar este aspecto, deve-se passar à modelagem do comportamento dinâmico das arquiteturas operacionais. Um dos paradigmas que possibilitam a modelagem e a simulação do comportamento dinâmico de sistemas complexos são as Redes de Petri (RdP) coloridas e temporizadas.

A coloração possibilita a construção de modelos genéricos nos quais a estrutura das sub-redes (seus lugares, transições, arcos e marcas) é independente das arquiteturas propostas.

Por exemplo, a estrutura do modelo dinâmico de uma rede de comunicação deve ser independente do número de equipamentos conectados a esta. São os parâmetros de coloração das marcas que tornam possível adaptar o modelo a uma determinada arquitetura [MEU97]. A

temporização, por sua vez, é indispensável para avaliar o desempenho temporal do sistema a partir de critérios como tempos de resposta, entre outros.

Neste trabalho será utilizado Design/CPN, programa que permite a edição e a simulação de RdP coloridas e temporizadas. Além disso, Design/CPN possibilita uma modelagem modular e hierárquica de sistemas, o que permite que se construa o modelo de uma arquitetura operacional através do acoplamento de modelos genéricos de tipos de equipamento e de tratamento.

Assim, durante a modelagem, pode-se partir dos níveis de abstração mais baixos do sistema, construindo modelos dinâmicos de cada tipo de componente, que descrevem seu funcionamento detalhado. Como a coloração possibilita que cada tipo de componente possa ser modelado por meio de sub-redes genéricas, os componentes individuais podem ser acoplados entre si por meio de lugares dedicadas de entrada e saída [BOT93]. Desta forma, constroem-se gradualmente modelos mais complexos, até o nível de abstração da arquitetura operacional como um todo (Figura 14), que é simulado segundo diferentes condições de operação. Na Figura 14.b, os quadrados são transições contendo sub-redes hierarquicamente inferiores em seu interior. Estas transições, chamadas em Design/CPN de *transições de substituição*, estão acopladas entre si por lugares de entrada e saída, formando o modelo completo e simulável da arquitetura de comando do sistema.

A metodologia apresentada até este ponto é uma metodologia multi-modelos, pois se utiliza de modelos tão diferentes como :

- SADT (diagramas de fluxo de dados) para modelar as características de processamento lógico das entradas e saídas do sistema;
- GRAFCET para modelar o comportamento seqüencial destas entradas e saídas;
- Redes de Petri para modelar a dinâmica das funções do sistema e o comportamento básico de cada equipamento de comando.

Neste ponto o leitor pode se questionar sobre a necessidade de utilizar formalismos tão diferentes, já que as Redes de Petri são por si só um paradigma capaz de modelar todos estes aspectos do comando do sistema. Existem vários trabalhos na bibliografia estabelecendo relações entre as RdP e outras abordagens de modelagem conceitual de sistemas.

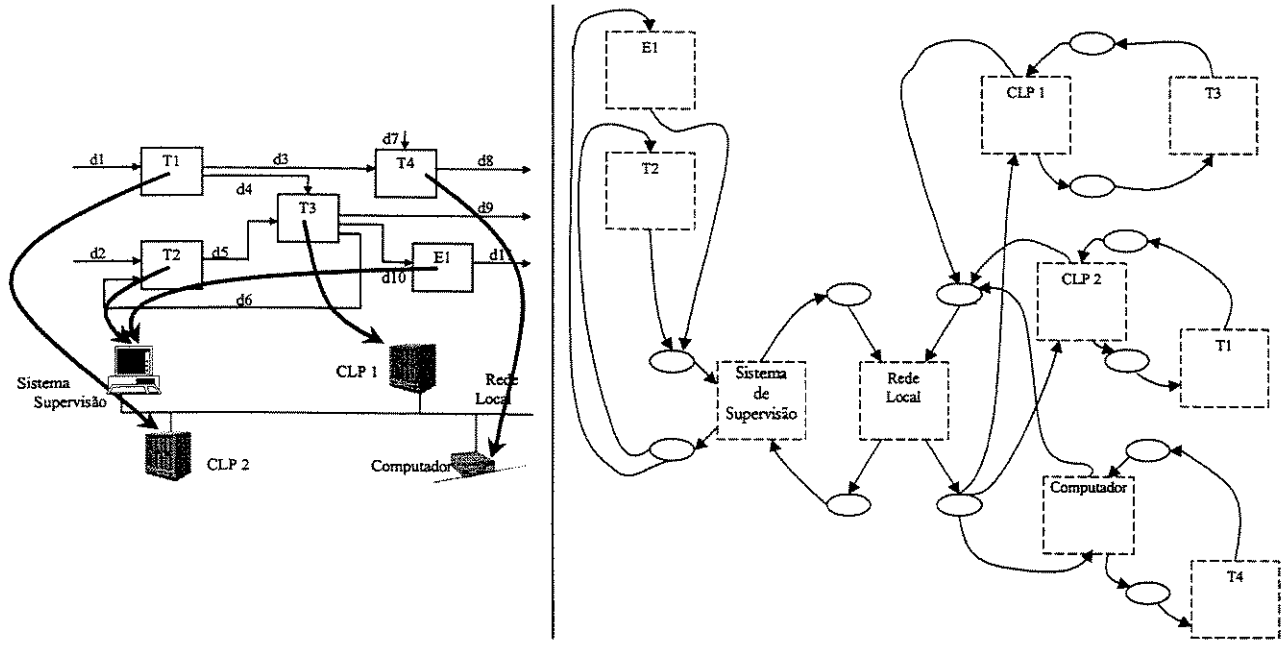


Figura 14. (a) Arquitetura operacional; (b) modelo dinâmico de arquitetura operacional

Citam-se como exemplos a comparação entre *diagramas de fluxo de dados* e RdPs do tipo *canal/atividade* (em contraposição às RdP do tipo *condição/evento*) e entre as RdP e modelos *entidade/relação* [HEU90]. As RdP também já foram utilizadas com sucesso para modelar o comando seqüencial de sistemas, por meio de modelos equivalentes ao GRAFCET (este próprio originalmente criado a partir de RdP *seguras*) [MUR95] e a diagramas lógicos Ladder [VEN95].

Explica-se a opção por uma metodologia multi-modelos pelo fato de que este trabalho destina-se a fornecer ferramentas **práticas** para o projeto de uma arquitetura de comando. Apesar das RdP poderem modelar a estrutura lógica de um sistema, é inegável que os diagramas de fluxo de dados são muito mais legíveis e adequados à representação deste aspecto do sistema. Da mesma forma, a implementação de uma RdP como modelo de controle seqüencial é muito mais trabalhosa do que representar a evolução de entradas e saídas do sistema por meio de diagramas do tipo estado-transição, como o GRAFCET.

Asim, uma metodologia multi-modelos assegura que cada aspecto do projeto da arquitetura de comando seja modelado da forma mais natural possível.

6.2. Aplicação a um Estudo de Caso

6.2.1. Estudo da Arquitetura de Comando de uma Aplicação Suporte: A Plataforma PIPEFA/PEREGA

Como exemplo de um sistema de produção concreto, utilizou-se a plataforma **PIPEFA/PEREGA (Plataforma Industrial para Pesquisa, Ensino e Formação em Automação / *Plate-forme Expérimentale pour la Recherche et l'Enseignement en Génie Automatique*)**.

PIPEFA/PEREGA é o fruto da colaboração entre o Laboratório de Automação Integrada e Robótica da Faculdade de Engenharia Mecânica da UNICAMP e o *Laboratoire d'Ingénierie Intégrée des Systèmes Industriels* do CESTI-ISMCM de Toulon, França, no projeto de pesquisa e transferência tecnológica Métodos e Ferramentas em Automação Industrial e Produção para o Desenvolvimento da Qualidade e Produtividade nas PME/PMI.

Trata-se de um SAP didático, na forma de célula de manufatura, implantado nestas duas instituições. Escolheu-se um de seus postos de trabalho, o *posto de montagem central*, como objeto de trabalho.

Visando modelar seu comportamento dinâmico e definir quais as medidas de desempenho temporal a serem coletadas, procedeu-se um estudo de seus pré-requisitos funcionais e do funcionamento de sua parte operativa, além dos equipamentos que constituem a arquitetura material de sua parte de comando, como redes de chão de fábrica e redes de comunicação de controladores lógicos programáveis (CLP).

6.2.2. Modelagem das Diferentes Arquiteturas Operacionais

Nesta etapa, adaptou-se o processo de projeto de arquiteturas de comando descrita nos

capítulos anteriores. Baseando-se nas especificações do posto de montagem estudado, uma arquitetura funcional foi proposta. Em seguida, as funções especificadas foram distribuídas segundo diferentes configurações nos equipamentos de sua parte de comando, como CLPs ou placas programáveis dedicadas à identificação dinâmica de etiquetas magnéticas. Assim, foram geradas diferentes arquiteturas operacionais teoricamente capazes de satisfazer os pré-requisitos do sistema.

6.2.3. Avaliação de Diferentes Configurações através de Design/CPN

Para avaliar as diferentes arquiteturas operacionais propostas, foi necessário construir modelos mais detalhados do funcionamento de cada componente material e das operações lógicas efetuadas por cada função de tratamento da parte de comando do posto escolhido. Utilizando-se Design/CPN (programa de edição e simulação de RdP coloridas e temporizadas), procedeu-se à construção de modelos dinâmicos das diversas partes do sistema. Estes puderam então ser acoplados segundo cada arquitetura operacional proposta e em seguida pôde-se simular vários ciclos de produção de PIPEFA/PEREGA.

Com a implementação de indicadores de desempenho (por exemplo, contadores ou variáveis para medir a evolução do tempo) associados a determinadas transições do modelo, os resultados desejados podem ser automaticamente gerados por Design/CPN sob forma gráfica.

6.3. Conclusão

Para atingir os resultados relativos à avaliação de desempenho de arquiteturas de comando, foi necessário um processo que consistiu em estudar uma determinada aplicação, passando pela modelagem de sua estrutura até a simulação dos modelos construídos. Os capítulos seguintes apresentam cada uma dessa fases em detalhe.

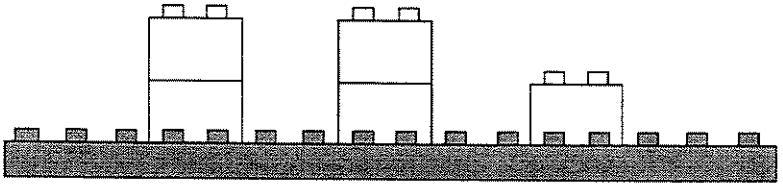
Capítulo 7

PIPEFA/PEREGA: Especificações e Funcionamento

A Plataforma Industrial para Pesquisa, Ensino e Formação em Automação, PIPEFA (e sua equivalente francesa PEREGA – *Plate-forme Expérimentale pour la Recherche et l'Enseignement en Génie Automatique*) é um SAP com finalidade pedagógica para o ensino de diversas competências em automação, servindo de suporte experimental para pesquisa nessa área.

7.1. Apresentação Geral

PIPEFA/PEREGA é uma célula de manufatura que realiza a montagem de peças *LEGO®*. O motivo da utilização dessas peças é seu baixo custo e a boa precisão mecânica. O exemplo típico de produto final deste SAP didático é uma peça de base, a *placa*, sobre a qual são montadas peças de menor tamanho, os *cubos*, como mostra a Figura 15.



Figurea15. Exemplo de produto final de PEREGA.

Os cubos podem ser montados sobre uma placa em três posições diferentes no eixo horizontal e no máximo em dois níveis no eixo vertical, o que totaliza seis posições possíveis.

Considerando-se apenas a presença ou a ausência de cubos nestas posições, a plataforma têm capacidade de montar 27 produtos diferentes. Ao adicionar a possibilidade de variação nas cores dos cubos, a gama de produtos da plataforma torna-se muito maior. Por exemplo, se considerássemos cubos de cores amarelo, vermelho e azul, a plataforma teria uma lista de 2197 produtos diferentes, o que potencialmente daria origem a diversas soluções concebíveis de gestão e ordenamento de produção.

Para realizar a montagem de produtos a partir de sua matéria-prima (cubes e placas), a plataforma é formada por quatro *postos de trabalho* e um *sistema de transferência.* A Figura 16 mostra a organização física da plataforma PIPEFA/PEREGA:

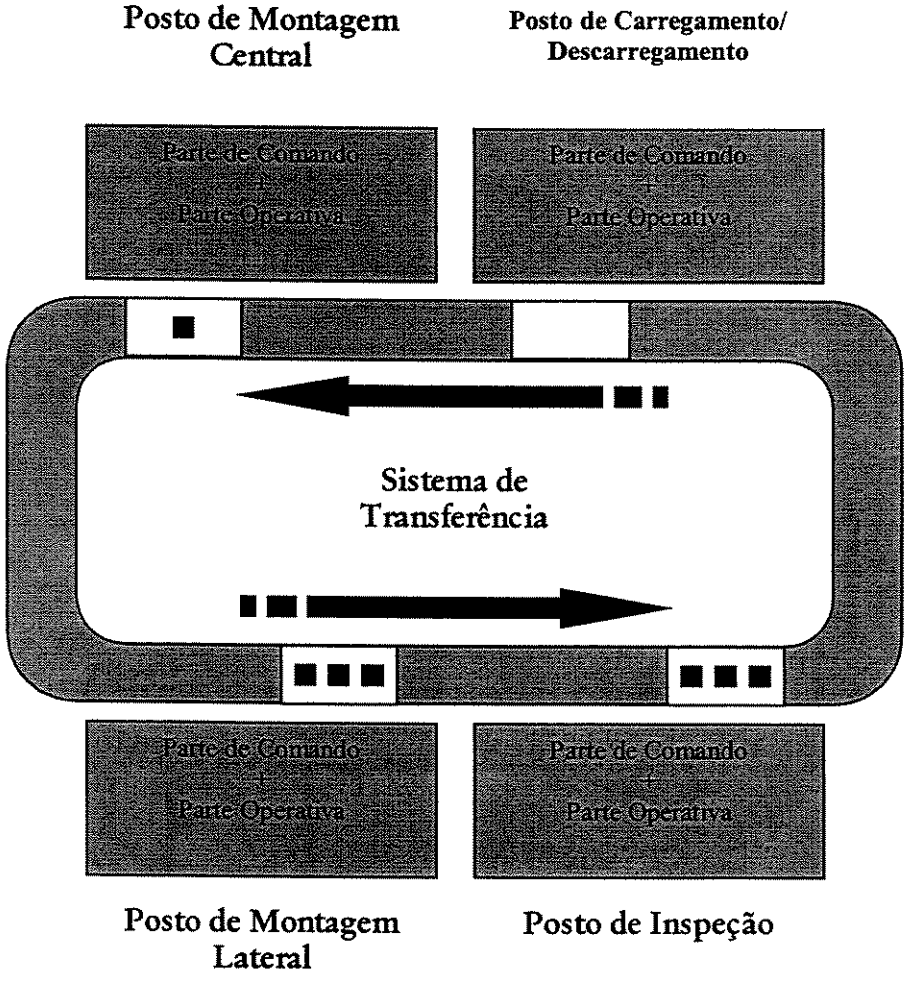


Figura 16. Organização de PIPEFA/PEREGA.

- o **sistema de transferência** consiste em um conjunto de esteiras rolantes ligando entre si os postos de trabalho; ele executa o transporte dos produtos entre os postos;

- o **posto de carregamento/decarregamento** é responsável por colocar placas vazias no sistema e por evacuar os produtos acabados;

- o **posto de montagem central** executa a montagem de cubos unicamente sobre a posição central das placas;

- o **posto de montagem lateral** monta cubos sobre as duas posições laterais da placa;

- o **posto de controle de qualidade**, no final do ciclo de montagem de um produto, ou seja, quando uma placa já passou pelos dois postos de montagem, é responsável por avaliar a conformidade entre o produto efetivamente montado e a especificação do produto que deveria ser montado, dentre a gama possível de produtos.

Este controle de qualidade está ligada ao processo de identificação de produtos. Cada placa lançada na linha de produção porta uma identificação que, por sua vez, informa os postos de trabalho do tipo de produto especificado para esta placa e, conseqüentemente, das operações que devem ou não ser efetuadas pelo respectivo posto de trabalho, de acordo com suas funções. Chamar-se-á o conjunto de dispositivos encarregados do processamento desta identificação, nos diferentes postos de trabalho, de ***sistema de identificação do produto***. Este sistema será apresentado mais detalhadamente à frente.

7.2. O Posto de Montagem Central

7.2.1. Parte Operativa e Funcionamento

A estrutura da parte operativa do posto de montagem central é mostrada na Figura 17. São atuadores com acionamento pneumático monoestáveis. O funcionamento do posto pode ser resumido pela seguinte seqüência de operações:

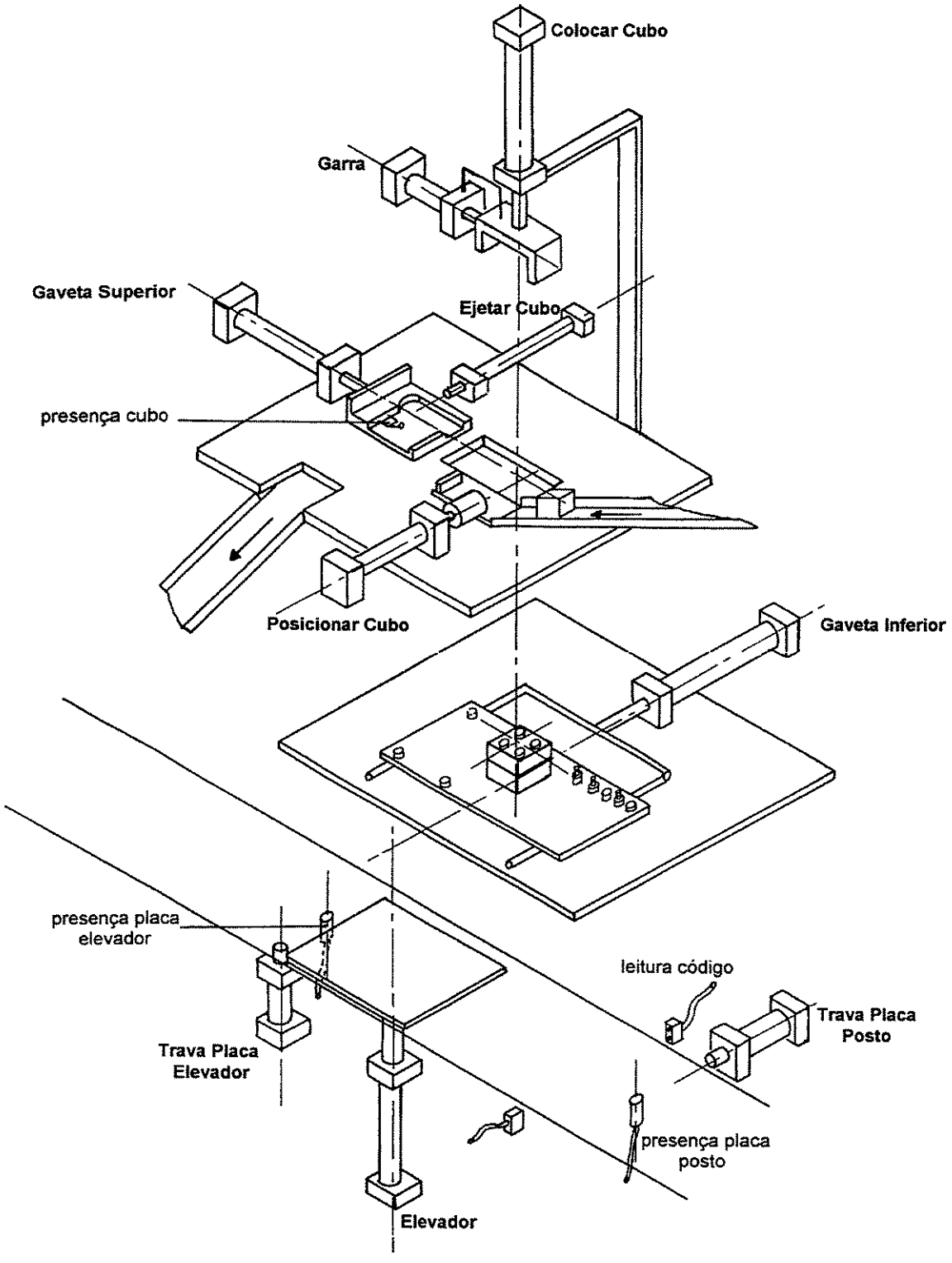


Figura 17. Parte operativa do posto de montagem central.

1. A placa chega ao posto através do sistema de transferência e sua identificação é lida;

2. A placa é em seguida carregada pelo atuador *Elevador* e levada até o interior do posto pelo atuador *Gaveta Inferior*;

3. O conjunto de atuadores da parte superior encarrega-se de manipular os cubos previamente estocados no posto (atuadores *Gaveta superior* e *Posicionar Cubo*) e de montá-los (atuador *Colocar Cubo*) sobre a placa que repousa sobre *Gaveta Inferior*;

4. Executando a operação inversa, de evacuação da placa, o atuador *Gaveta superior* coloca a placa já montada sobre *Elevador*, que a desce finalmente até o sistema de transferência.

7.2.2. Parte de Comando

A parte de comando dos postos de trabalho consiste de CLPs (CLPs Telemecanique®, modelos TSX37 e TSX17). Estes são ligados entre si através de rede UNITELWAY. Além disso, a arquitetura material do comando pode comportar placas programáveis dedicadas, como por exemplo, para desempenhar o papel de *sistema de identificação de produtos*.

O Sistema de Identificação de Produtos em PIPEFA/PEREGA se dava pela leitura de pequenos pinos afixados sobre a placa. A presença ou ausência de um pino em determinadas posições era detectada por um sensor de fibra ótica situado na entrada de cada posto de trabalho. Um CLP era responsável pelo tratamento lógico destes sinais e, portanto, pela identificação do produto a ser montado.

Recentemente, o ISMCM-CESTI de Toulon adquiriu um sistema de identificação por meio de etiquetas magnéticas, produzido pela companhia *BALOGH®*. Em sua totalidade, este sistema é composto dos seguinte equipamentos:

- uma **placa programável** centralizadora do sistema, responsável pela lógica do código de identificação e capaz de automatizar alguns processos graças a 16 entradas e 16 saídas digitais;

- **etiquetas magnéticas regraváveis**, com capacidade de memória de 96 bytes cada uma, na qual deve ser armazenado o código relativo ao produto de uma determinada placa;
- **contatos de leitura/gravação** (um contato para leitura e outro para gravação em cada posto de trabalho), para acessar a memória das etiquetas;
- quatro **módulos de diálogo**, um para cada posto de trabalho, responsáveis pela transmissão do código entre a placa programável e os contatos de leitura/escrita de um posto.

A comunicação entre a placa programável e os módulos de comunicação é feita através de uma rede de chão de fábrica de protocolo CAN (*Control Area Network*).

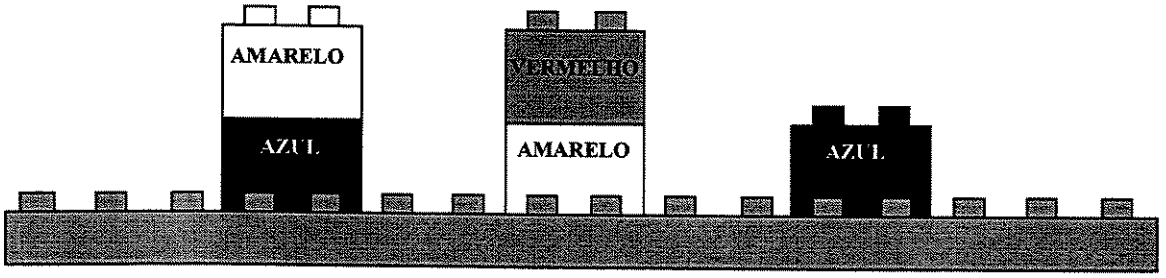
A capacidade de memória das etiquetas (96 bytes) possibilita a codificação e a gestão de diferentes cores para os cubos, o que, como citado anteriormente, aumentaria consideravelmente a gama de produtos de PIPEFA/PEREGA. Além disso, como as etiquetas podem ser alternativamente lidas ou gravadas, é possível rastrear a seqüência e operações sobre uma placa sem a presença obrigatória de um sistema de supervisão.

Uma opção simples de codificação é mostrada na Figura 18. Neste caso, os cubos podem ser de cor azul, amarelo ou vermelho. Utilizar-se-iam seis bytes, um para cada cubo. Em cada byte, os três primeiros bits seriam utilizados somente para leitura (bits 0, 1 e 2 para as cores azul, amarelo e vermelho, respectivamente) e os três bits seguintes somente para gravação (da mesma forma, bits 3, 4 e 5 para as cores azul, amarelo e vermelho, respectivamente). Por meio dos bits de leitura, a placa programável comunica o CLP que comanda um posto de trabalho do código do produto a ser montado e conseqüentemente, das operações que ele deve executar. De forma recíproca, as operações no posto (identificadas pelos sinais dos sensores) fazem com que o CLP comunique a placa da seqüência de valores a ser gravada nos bits de gravação da etiqueta magnética.

Este sistema, por etiquetas magnéticas, foi implantado sem que se suprimisse o sistema anterior, por pinos. Assim, PIPEFA/PEREGA pode funcionar alternativamente com os dois sistema de identificação do produto.

Como a gravação e a leitura das etiquetas magnéticas não podem ser realizadas com a placa em movimento, a única modificação feita ao fluxo dos produtos foi a adição de um atuador

suplementar destinado a parar a placa (que será chamado de *Trava Placa Leitura*) para a leitura da etiqueta magnética pelo contato de leitura; a parada para gravação já sendo assegurado pelo atuador *Trava Placa Elevador*, como mostra a Figura 19.



			Gravação			Leitura		
Cores	-	-	V	Am	Az	V	Am	Az
	0	0	0	1	0	0	1	0
	0	0	1	0	0	1	0	0
	0	0	0	0	1	0	0	1
	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	1
	0	0	0	1	0	0	1	0
Bits	7	6	5	4	3	2	1	0

Cubo Central Inferior
Cubo Central Superior
Cubo Direito Inferior
Cubo Direito Superior
Cubo Esquerdo Inferior
Cubo Esquerdo Superior

Figura 18. Exemplo de codificação de um produto.

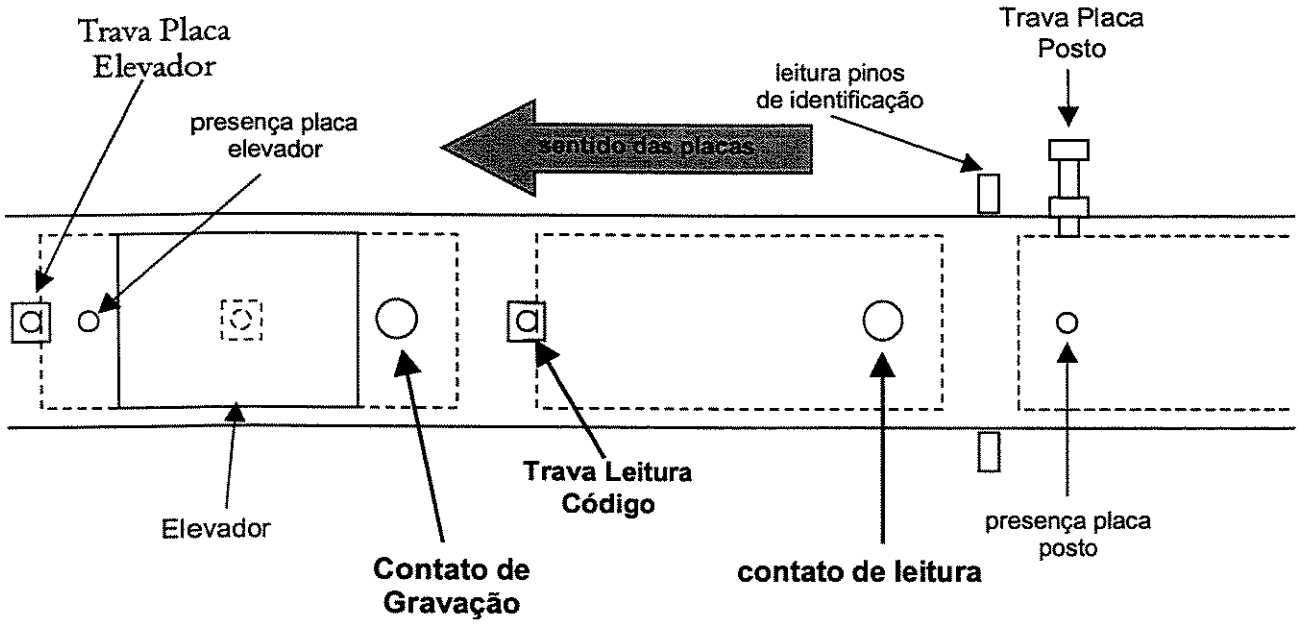


Figura 19. Sistema de transferência no posto de montagem central: configuração para identificação do produto por meio de etiquetas magnéticas.

Capítulo 8

Modelagem de Diferentes Configurações de Comando

8.1. Arquitetura Funcional

A partir dos pré-requisitos funcionais da plataforma PIPEFA/PEREGA, é possível modelar a arquitetura funcional do posto de montagem central. Esta modelagem foi feita através do método SADT, o que possibilita construir de maneira estruturada os modelos necessários.

8.1.1. Sintaxe de SADT

SADT (Structured Analysis and DesignTechnique) foi desenvolvido por D. T. Ross e colegas na Softech Inc.. SADT engloba uma linguagem gráfica e um conjunto de métodos e diretrizes para usá-la. Um modelo SADT consiste de um conjunto ordenado de diagramas de *análise estruturada*. Cada diagrama é feito sobre uma página, e deve conter de três a seis caixas (retângulos) além de arcos conectando-as entre si. A Figura 20 ilustra o formato de uma caixa de um actigrama (o tipo de diagrama SADT mas comum: as caixas representam atividades e os arcos representam o fluxo de dados entre as atividades).

É importante notar que quatro tipos distintos de arcos podem ser acoplados a cada caixa. Arcos entrando pelo lado esquerdo contêm entradas e arcos partindo do lado direito da caixa são as saídas. Arcos entrando por cima representam **fluxos de controle** e os arcos entrando por baixo especificam o **mecanismo que dá substrato** à atividade em questão. Os conceitos de *entrada*, *saída*, *controle* e *mecanismo* definem o contexto de cada caixa em um diagrama SADT.

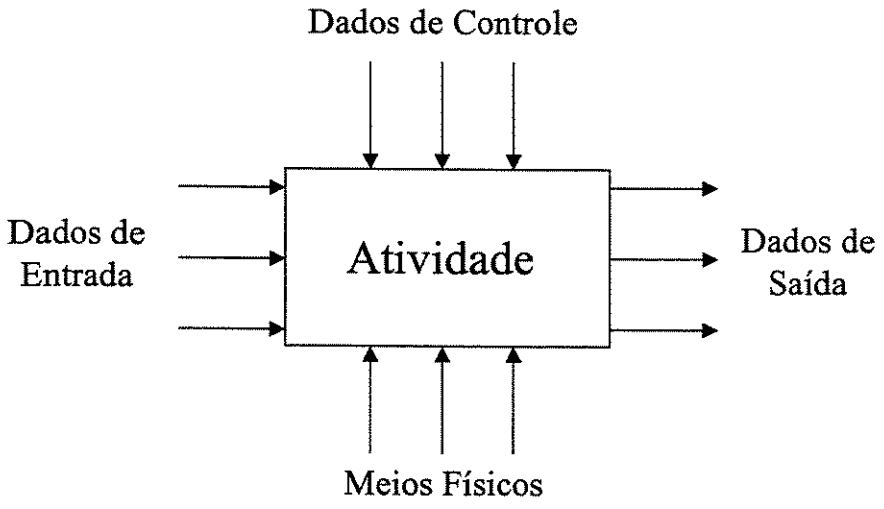


Figura 20. SADT – Sintaxe de uma caixa de atividades.

Em um actigrama, as entradas e saídas são fluxos de dados, e os mecanismos são pocessadores (dispositivos ou pessoas). Controle é um dado que é usado mas não é modificado pela atividade.

Os arcos podem ser bidirecionais, indicando reciprocidade de entrada/saída entre caixas. Nesses casos, se um arco de saída de uma caixa entra em sob a forma de arco de controle, existe outro arco, de sentido inverso, que é saída deste último e arco de controle para o primeiro. Para indicar o caráter bidirecional de arcos em diagramas muito complexos, SADT sugere que se coloque um pequeno ponto em cada extremidade do arco bidirecional.

8.1.2. Os modelos de arquitetura funcional

As figuras 21 e 22 mostram os actigramas A-1 e A-0, que possibilitam localizar hierarquicamente o sistema estudado. As funções que devem ser executadas pela parte de comando do posto são representadas no actigrama A0 na Figura 23.

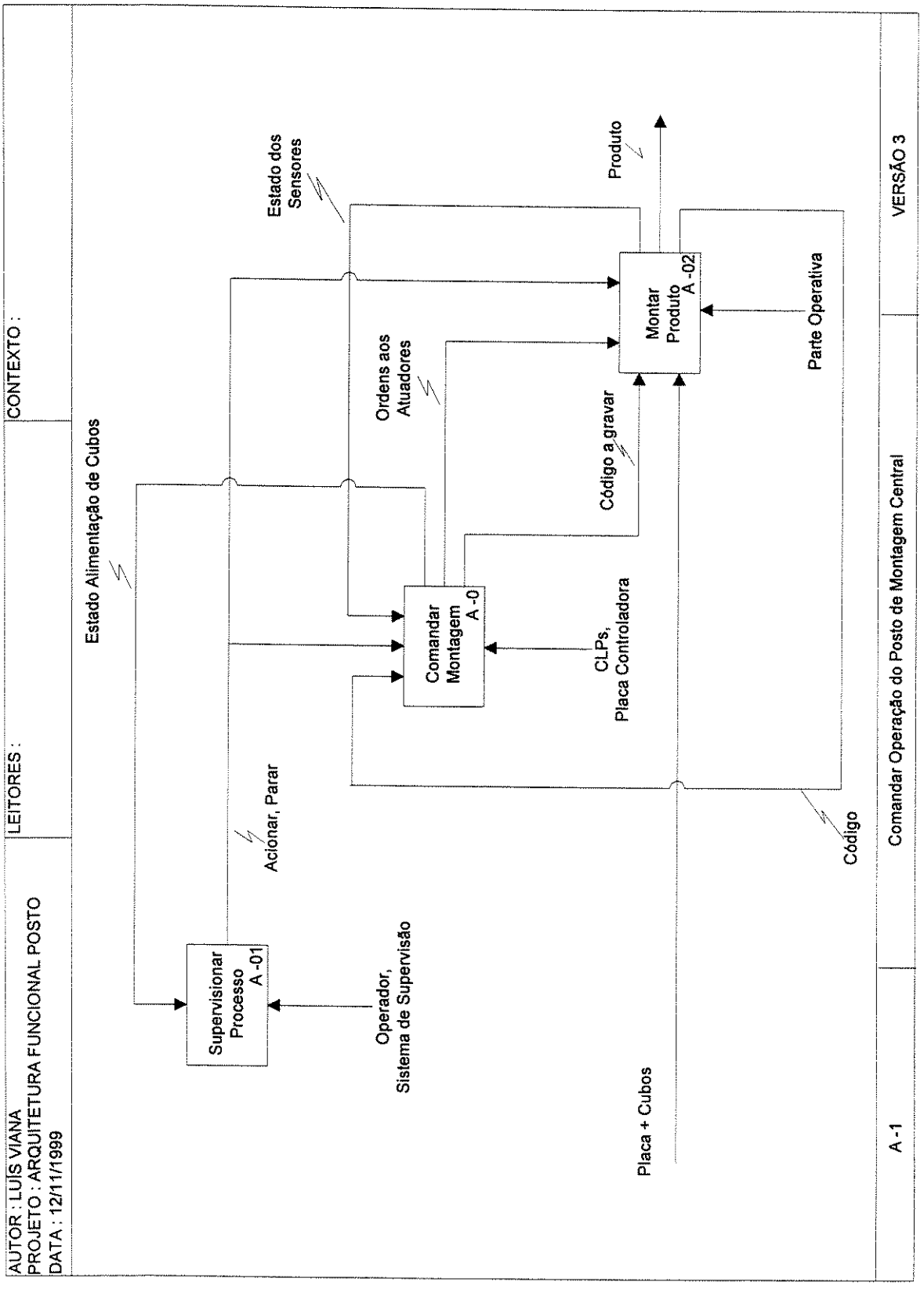


Figura 21. Actigrama de contexto A-1.

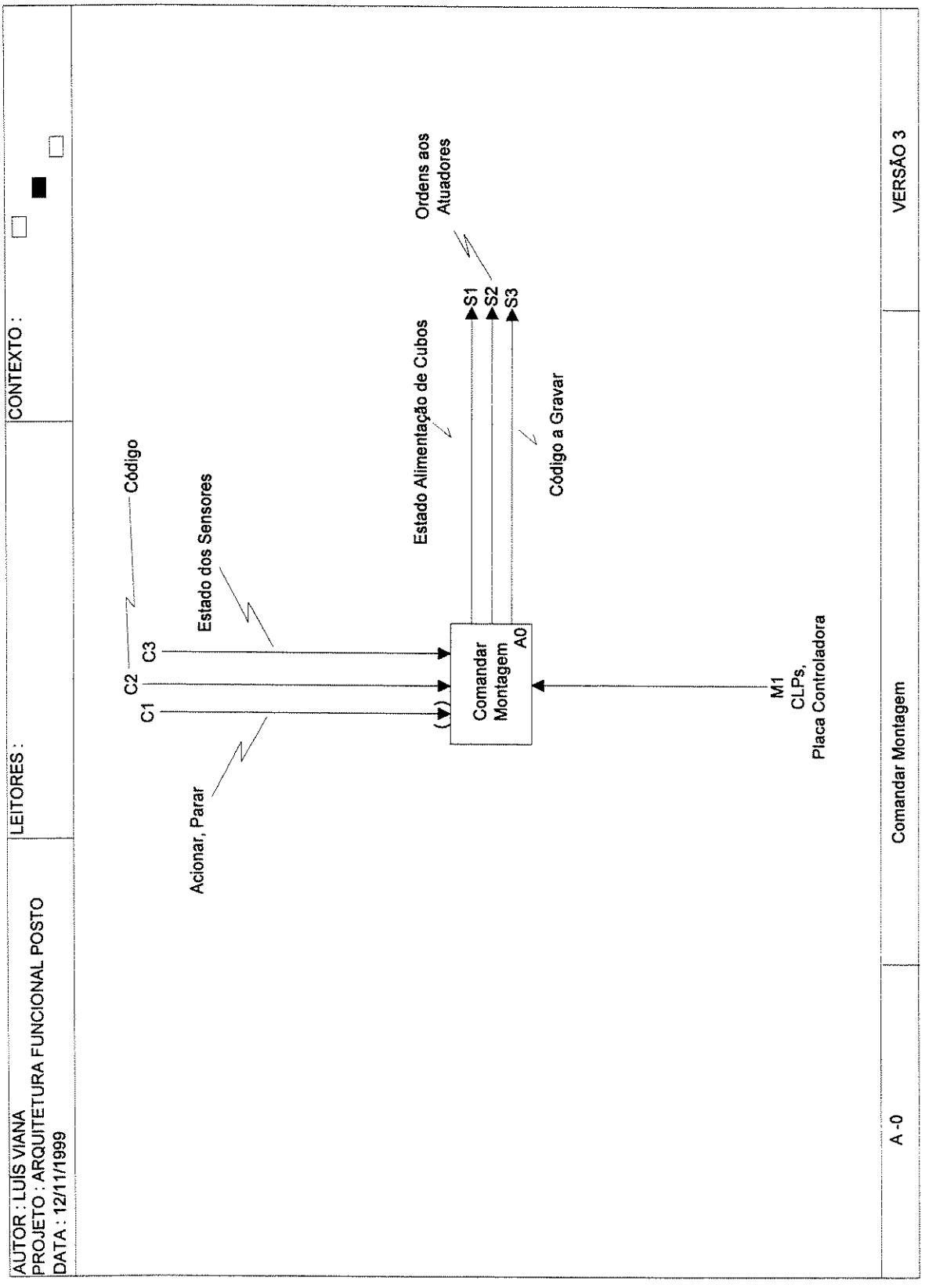


Figura 22. Actigrama A-0.

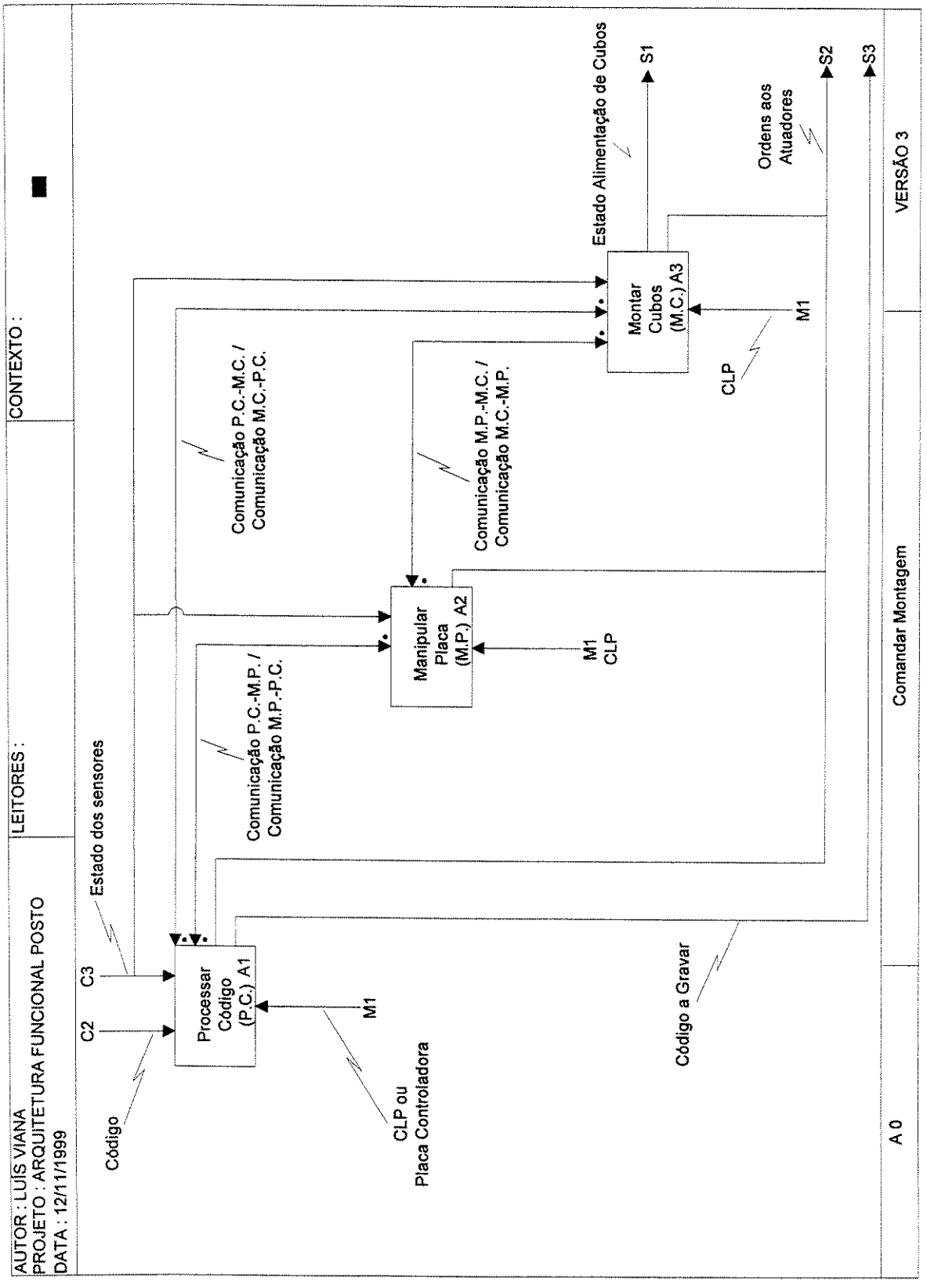


Figura 23. Arquitetura funcional do posto de montagem central: modelo funcional.

GLOSSÁRIO PARCIAL 1

Comunicação P.C.-M.P. = *mensagem da função Processar Código para a função Manipular Placa*

Comunicação P.C.-M.C. = *mensagem da função Processar Código para a função Montar Cubos*

Comunicação M.P.-P.C. = *mensagem da função Manipular Placa para a função Processar Código*

Comunicação M.P.-M.C. = *mensagem da função Manipular Placa para a função Montar Cubos *

Comunicação M.C.-P.C. = *mensagem da função Montar Cubos para a função Processar Código*

Comunicação M.C.-M.P. = *mensagem da função Montar Cubos para a função Manipular Placa*

Estado dos Sensores = **

[SENPOSTO *presença placa posto*|
SENELEV *presença placa elevador*|
GRAVCONF*confirmação gravação*|
ELEV1 *elevador OUT|
ELEV0 *elevador IN*|
GAVINF1 *gaveta inferior OUT*|
GAVINF0 *gaveta inferior IN*|
GAVSUP1 *gaveta superior OUT*|
GAVSUP0 *gaveta superior IN*|
POSCUB1 *posicionar cubo OUT*|
POSCUB0 *posicionar cubo IN*|
COLCUB1 *colocar cubo OUT*|
COLCUB0 *colocar cubo IN*]

Ordens aos Atuadores = **

[TRAVALEIT	*Trava Placa Leitura*|
LEIT	*Leitura*|
REQLEIT	*Requisição de Pacote de Leitura*|
TRAVAELEV	*trava Placa Elevador*|
REQGRAV	* Requisição de Confirmação de Gravação*|
ELEV	*Elevador Placa|
GAVINF	*gaveta Inferior*|
GAVSUP	*Gaveta Superior*|
POSCUB	*Posicionar Cubo*|
COLCUB	*Colocar Cubo*|
GARRA	*Garra*]

8.12. Os modelos de arquitetura funcional (*CONTINUAÇÃO*)

Através de A0 pode-se constatar que as funções principais da parte de comando do posto central são:

- processar o código de identificação do produto;
- acolher e evacuar a placa;
- montar os cubos necessários.

A função **Processar código** consiste em processar o código presente na placa, seja pelo sistema de pinos, seja pelo sistema de etiquetas magnéticas *BALOGH®*. Como, nesta segunda modalidade, tanto a gravação quanto a leitura das etiquetas necessitam que as placas sejam paradas, adicionou-se a esta função o **comando dos atuadores de parada (travas)** no sistema de transferência. Desta forma, esta função permanece relativamente homogênea nos dois tipos de utilização de PEREGA. A única diferença é a operação de gravação na placa quando o sistema *BALOGH®* é utilizado. Isto adiciona um novo dado ao modelo, denominado *código gravado*.

Manipular Placa significa comandar as operações alternadas dos atuadores *Elevador* et *Gaveta Inferior* (vistos na Figura 17), ou seja, o recebimento da placa e sua evacuação depois de montado o produto;

Montar Cubos é responsável pelos atuadores que realizam a montagem de cubos e pelos respectivos sensores.

A evolução temporal destas funções é a seguinte:

- a função *Processar Código*, depois de ter detectado a chegada de uma placa ao posto e lido o código do produto, emite simultaneamente mensagens para as funções *Manipular Placa* e *Montar Cubos*, ativando-as.
- *Manipular Placa* e *Montar Cubos* são interdependentes e trocam mensagens de sincronização entre si para assegurar a boa coordenação entre o posicionamento da placa no interior do posto e a montagem dos cubos;

• quando a montagem e a evacuação do produto terminam, *Manipular Placa* e *Montar Cubos* emitem mensagens de fim de operação para a função *Processar Código*. Se o sistema de codificação magnética é utilizado, *Processar Código* deve ainda assegurar a gravação do código proveniente das operações realizadas por *Montar Cubos* (considera-se que o código de gravação é adicionado ao pacote da mensagem de fim de operação de *Montar Cubos*) na etiqueta magnética, produzindo, então, o dado *código gravado*.

O comportamento de cada uma das funções em relação a suas entradas e saídas é descrito por um GRAFCET distinto. Por exemplo, a Figura 24 mostra o comportamento da função *Manipular Placa*. Os GRAFCETs individuais das funções de comando do sistema são apresentados da seção **ANEXO A**.

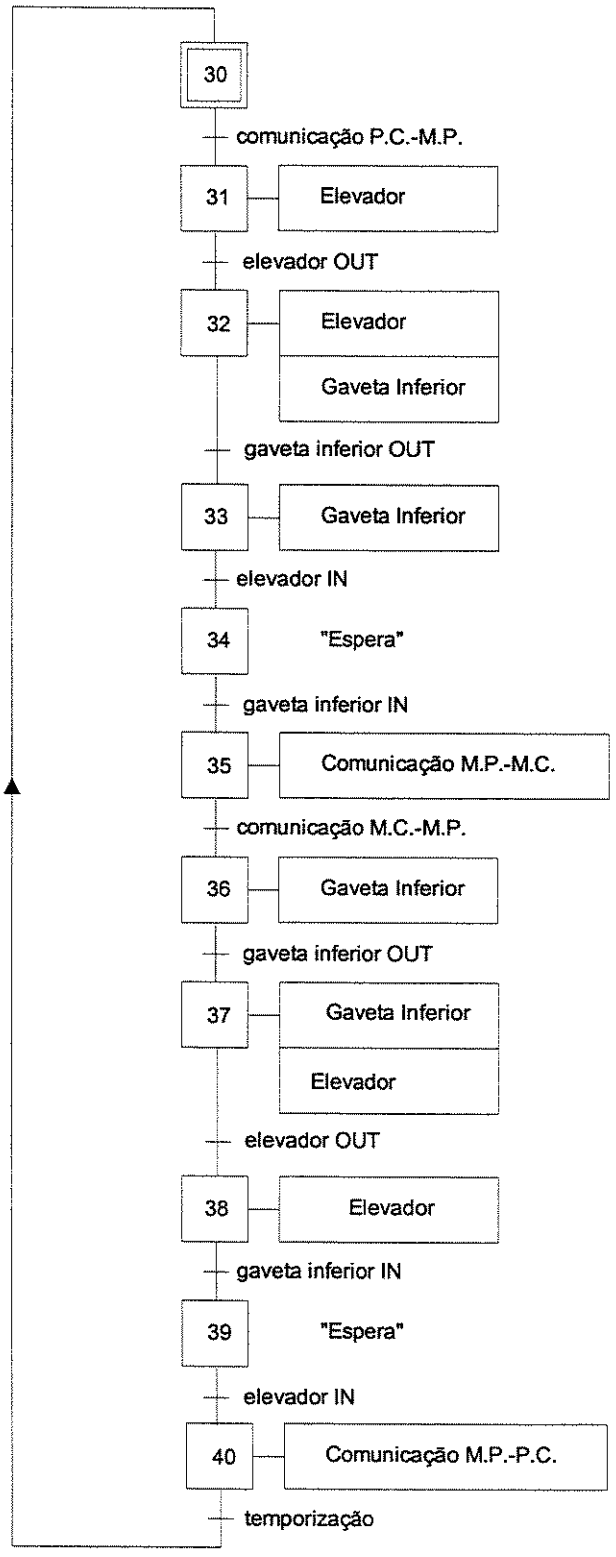


Figura 24. GRAFCET modelando o comportamento da função Manipular Placa.

8.2. Arquiteturas Materiais

Duas arquiteturas materiais são propostas para executar o comando do posto:

A primeira (Figura 25) consiste em três CLPs ligados em barramento por rede *UNITELWAY*. O protocolo *UNI-TE* de *UNITELWAY* estabelece relações do tipo mestre/escravo entre as entidades conectadas e permite o acesso ao controle do canal por *polling*.

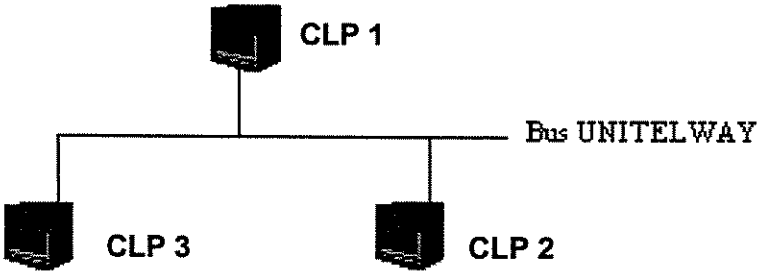


Figura 25. Primeira arquitetura material.

A segunda (Figura 26) é constituída por dois CLPs e pela placa progamável de codificação magnética *BALOGH®*, ligados em barramento por rede *UNITELWAY*. A placa é ligada a seus módulos de diálogo por rede de chão de fábrica de perfil CAN (*Controller Area Network*). A comunicação por protocolo CAN se dá por difusão (*broadcast diffusion*).

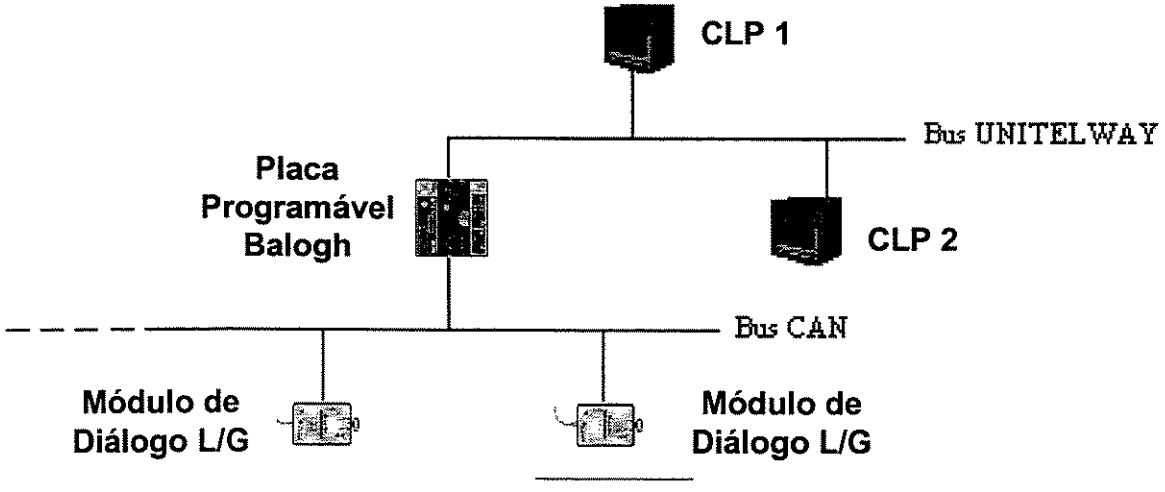


Figura 26. Segunda arquitetura material.

8.3. Arquiteturas Operacionais

As arquiteturas operacionais a serem avaliadas são obtidas através da projeção da arquitetura funcional sobre as arquiteturas materiais propostas.

Como a rede *UNITELWAY* apresenta uma estrutura hierárquica (um mestre e diversos escravos), é preciso também definir se uma função da arquitetura funcional será executada pela entidade mestre ou por uma das entidades escravas.

Portanto, para a assim chamada ***Arquitetura A*** (Figura 27), com três CLPs, escolheu-se executar:

- ***Processar Código*** no **CLP 3** como **mestre**;
- ***Manipular Placa*** no **CLP 1** como **escravo**;
- ***Montar Cubos*** no **CLP 2** como **escravo**.

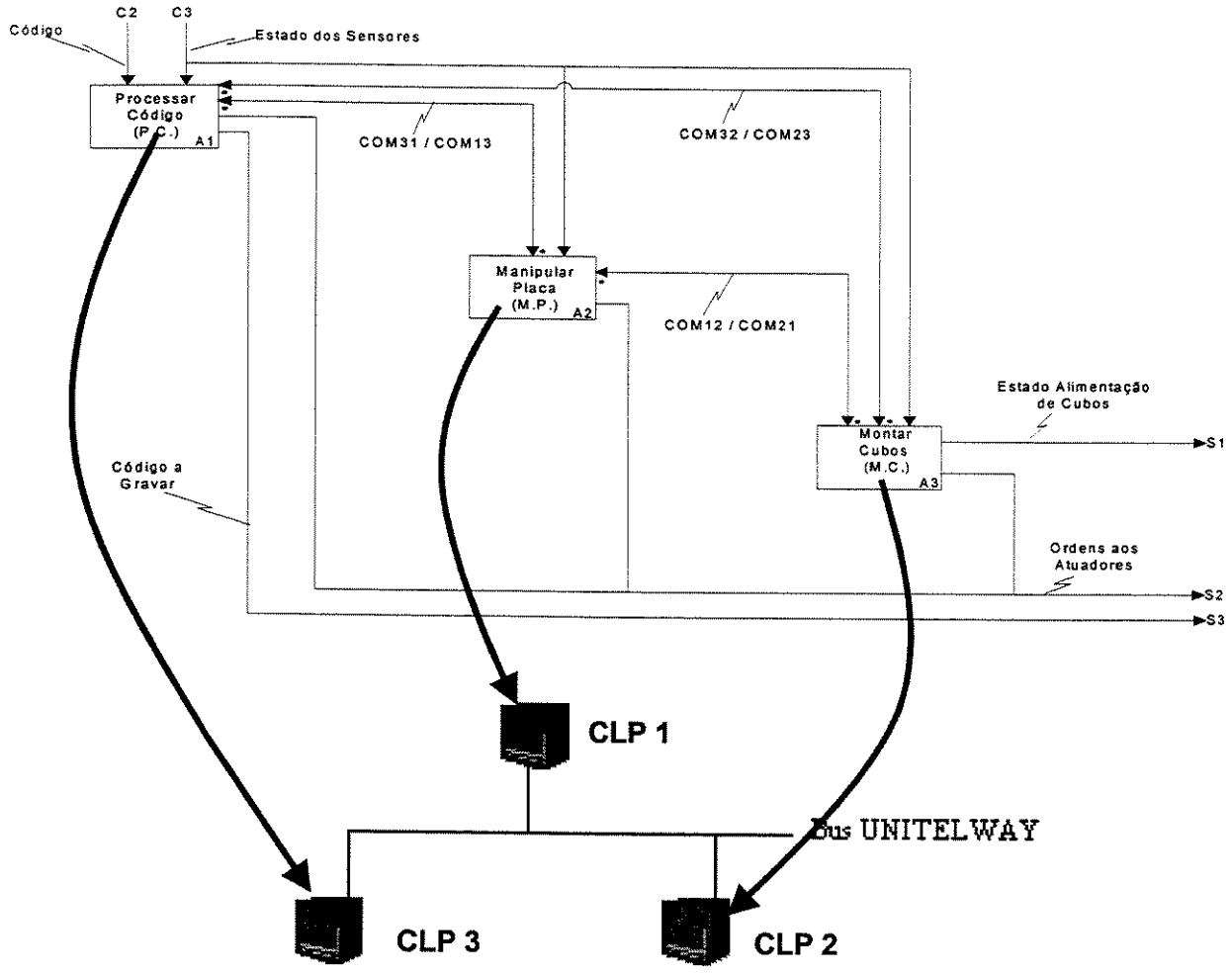


Figura 27. Arquitetura operacional A.

Como ***Arquitetura B*** (Figura 28), escolheu-se o CLP 2 como mestre da rede. Na verdade, a placa *BALOGH®*, quando ligada em rede *UNITELWAY*, somente pode funcionar em modo escravo.

Portanto, no caso da Arquitetura B, com dois CLP e a placa programável, a repartição é a seguinte:

- ***Processar Código*** na **placa programável** (entidade **3**) como **escravo**;
- ***Manipular Placa*** no **CLP 1** como **escravo**;
- ***Montar Cubos*** no **CLP 2** como **mestre**.

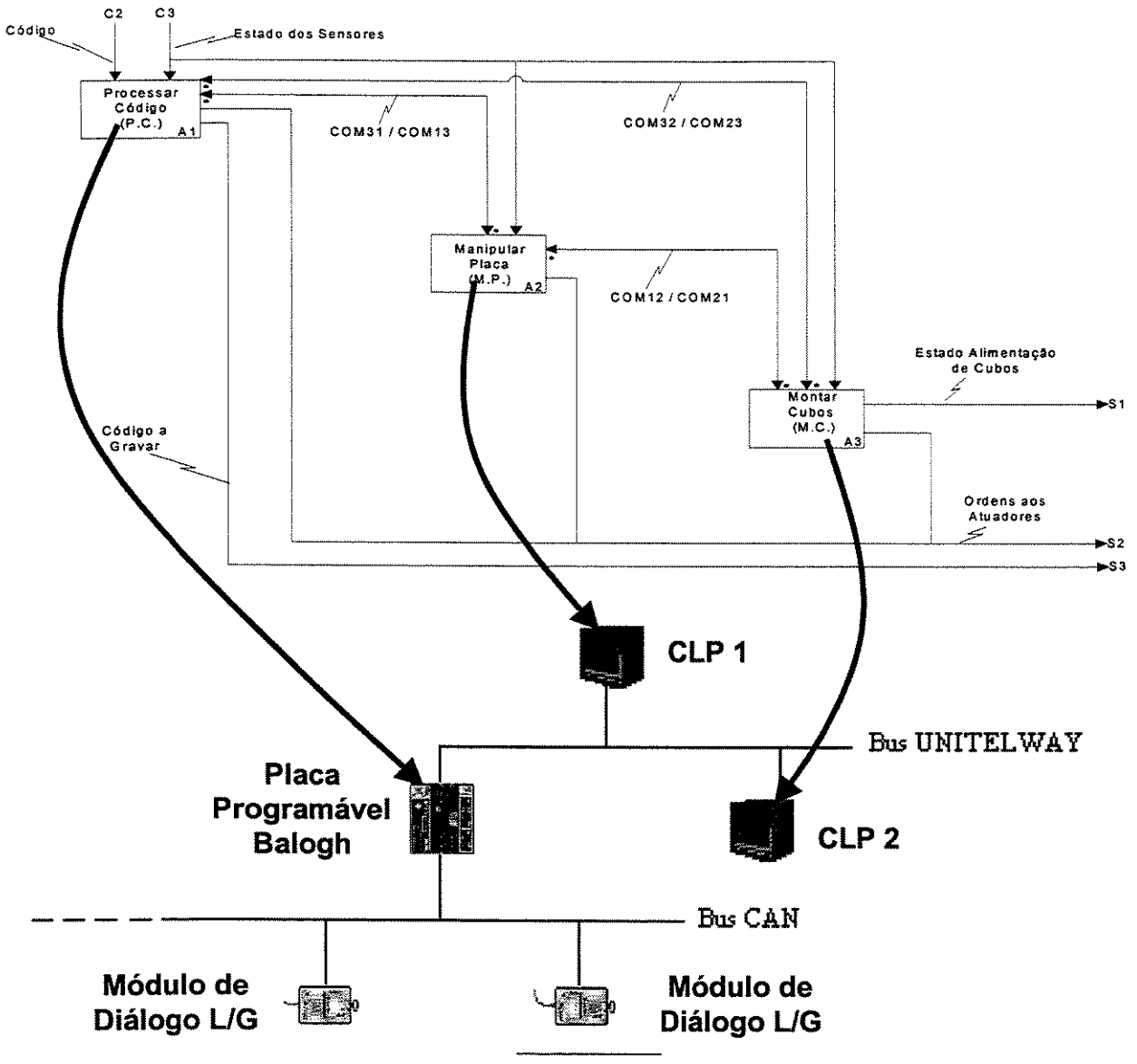


Figura 28. Arquitetura operacional B.

Os GRAFCETs que comandam cada um dos equipamentos, para as duas arquiteturas operacionais propostas, estão mostrados no **ANEXO B**. Por questões de simplificação, estes GRAFCETs partem das seguintes hipóteses:

- o período com que as placas chegam ao posto é significativamente superior ao tempo de um ciclo de produção do posto, ou seja, uma nova placa sempre chega ao posto após a partida da placa anterior;
- o resultado da leitura do código é sempre verdadeiro, ou seja, o código de todas as

placas indicam uma operação de montagem no posto;

- apenas um cubo é montado por placa.

Além disso, há diferença entre as entradas e saídas da função Processar Código de acordo com o sistema de identificação que é utilizado: ou por pinos ou por etiquetas magnéticas.

Além do comando dos diversos atuadores vistos anteriormente e das *receptividades* associadas às diversas transições (por exemplo, *gaveta inferior IN* ou *gaveta inferior OUT*), os GRAFCETs contêm etapas e transições destinadas à comunicação entre os equipamentos, através da troca de mensagens ente eles. Estas etapas e transições estão indicadas pela legenda "**COMXY**" (por exemplo, COM31, COM32, COM12, etc.), em que X e Y indicam os números correspondentes aos equipamentos em comunicação (X o emissor e Y o receptor).

Por exemplo, COM31 significa uma comunicação da entidade 3 para a entidade 1. Caso se trate da Arquitetura A, isto significa uma mensagem do CLP 3 para o CLP 1. No caso da Arquitetura B, trata-se de uma mensagem da placa programável *BALOGH®* (denominada *entidade 3*) para o CLP 1.

Considerar-se-ão as "COMXY" como variáveis booleanas internas aos equipamentos. Ou seja, a ação associada a uma etapa "COM12" pode ser considerada como *"emitir COM12"* enquanto uma transição de *receptividade* "COM23" pode ser lida como *"COM23 recebida"*. O Glossário Parcial 2 lista esses dados e a Figura 29 mostra um exemplo de comunicação entre CLPs na Arquitetura B.

GLOSSÁRIO PARCIAL 2

COM12 = *mensagem do equipamento 1 para o equipamento 2*

COM13 = *mensagem do equipamento 1 para o equipamento 3*

COM21 = *mensagem do equipamento 2 para o equipamento 1*

COM23 = *mensagem do equipamento 2 para o equipamento 3*

COM31 = *mensagem do equipamento 3 para o equipamento 1*

COM32 = *mensagem do equipamento 3 para o equipamento 2*

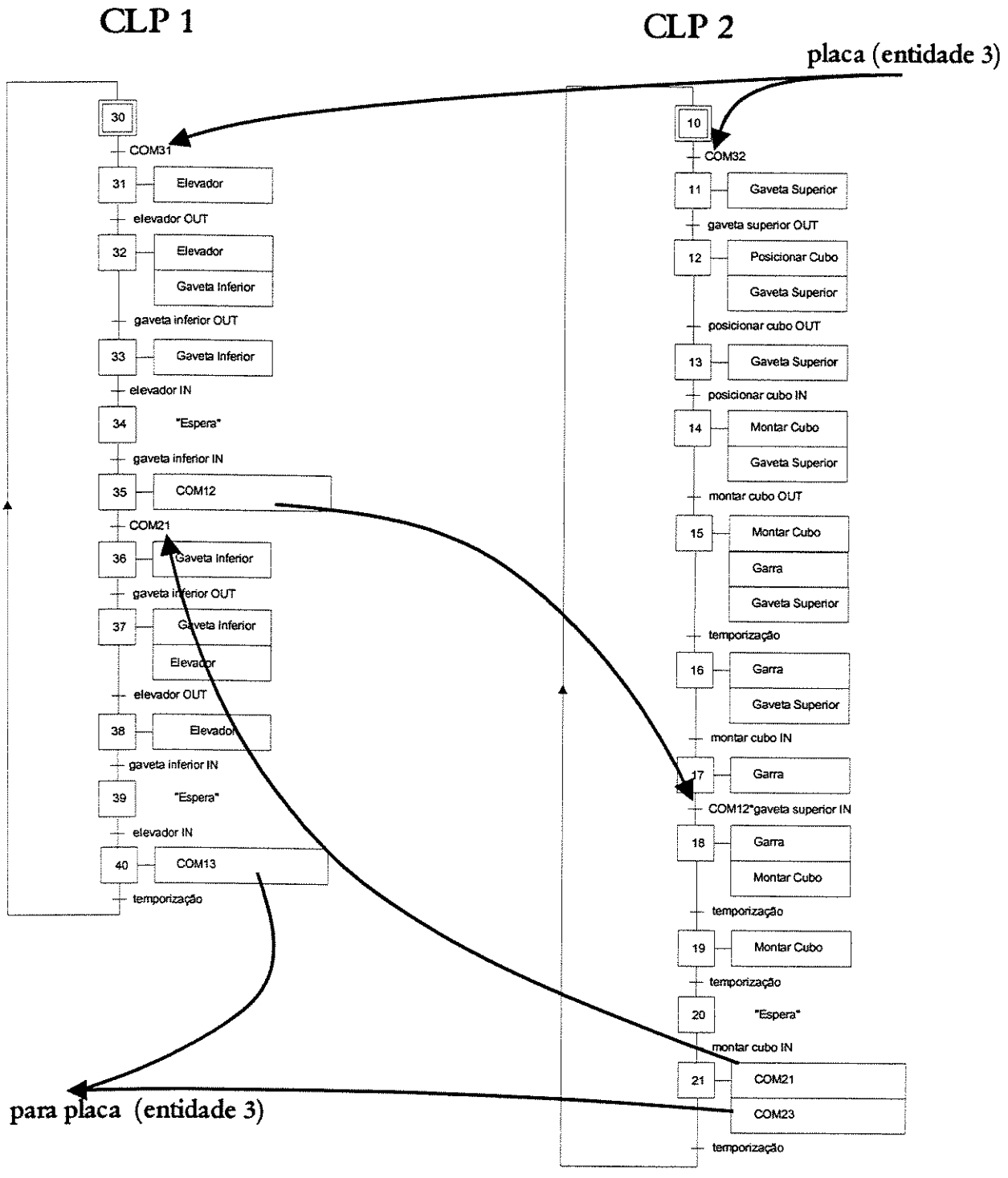


Figura 29. Exemplo de comunicação entre funções distribuídas em equipamentos

Capítulo 9

Avaliação das Arquiteturas Propostas

9.1. Os Indicadores de Desempenho

A eficácia da comunicação entre as diversas entidades será o principal critério para a validação e a avaliação das arquiteturas operacionais propostas.

Desta forma, os indicadores de desempenho aplicados serão os intervalos de tempo entre a emissão de uma mensagem por um equipamento e o comando da ação resultante da recepção desta mensagem em um outro equipamento.

Portanto, no caso das mensagens do tipo COMXY nos GRAFCETs do ANEXO B, serão medidos os intervalos entre o disparo da transição necessária à emissão de COMXY, em X, e a ação que se segue ao recebimento de COMXY, em Y. A Figura 30 mostra o caso de COM31 e COM32.

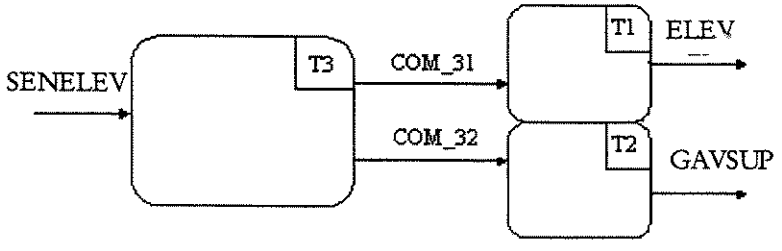


Figura 30. COM31 e COM32, mensagens do equipamento 3 para os equipamentos 1 et 2.

Por exemplo, para as duas arquiteturas propostas, A e B, a mensagem COM31 se situa entre

a recepção do sinal do sensor SENELEV (*presença placa elevador*) pelo equipamento 3 (CLP3 na Arquitetura A e placa programável na Arquitetura B) e a ativação de ELEV (*Elevador*) por parte do equipamento 1 (CLP 1). Propõe-se que se meça, então, o intervalo entre SENELEV e ELEV, provenientes de dois tratamentos (T3 e T1) fisicamente distribuídos, mas ligados via rede. Da mesma forma, o intervalo entre SENELEV e GAVSUP (*Gaveta Superior*), envolvendo a mensagem COM32, é um outro indicador de desempenho do sistema.

Considerando, então, a totalidade da comunicação entre os tratamentos das duas arquiteturas propostas, os indicadores de desempenho a serem medidos são:

- os intervalos entre os sinais **presença placa elevador** e **Elevador+Gaveta Superior**, resultantes de COM31 e COM32 (Figura 30);
- o intervalo entre **gaveta inferior IN** e **Colocar Cubo**, resultante de COM12;
- o intervalo entre **colocar cubo IN** e **Gaveta Inferior**, resultante de COM21;
- o intervalo entre **elevador IN** e um **sinal de fim de ciclo de montagem** (por parte do equipamento 3), resultante de COM13.

9.2. A Simulação em Design/CPN

Design/CPN apresenta boa ergonomia e características adequadas à modelagem e à avaliação de sistemas complexos. Uma descrição de seus principais aspectos é feita a seguir.

9.2.1. Características de Temporização

O tipo de temporização de Design/CPN é a *T-temporização* (temporização através das transições) com modificações que possibilitam associar temporizações diretamente aos diferentes arcos que partem de uma transição. Desta forma, a temporização gerada pelo disparo de uma transição pode ser diferente de acordo com os diferentes arcos de saída utilizados pelas marcas.

9.2.2. Os Modos de Simulação

Design/CPN possibilita a simulação de modelos através de dois modos distintos.

A **simulação interativa**, ou seja, passo a passo, permite que se observe graficamente, após o cada disparo de transição (passo de simulação), o novo estado para o qual o sistema evoluiu. Este modo é útil durante a construção de um modelo, sobretudo no que se refere a seu *debugging*.

No entanto, simular passo a passo modelos complexos exige um investimento considerável de tempo. A **simulação automática** permite simular um modelo por vários ciclos de maneira rápida, tornando possível a coleta de dados estatísticos. Neste caso, não é possível observar as mudanças de estado do modelo, pois sua representação gráfica é atualizada apenas no fim da simulação.

Em ambos os modos, Design/CPN oferece diversas opções para controlar os ciclos de simulação. Por exemplo, é possível determinar a duração ou pelo tempo passado no modelo ou pelo número de passos de simulação executados.

9.2.3. A Adição de Código de Programação

É possível adicionar código em linguagem CPN/ML às transições de um modelo. O código associado a uma transição é executado cada vez que esta transição é disparada. Isso possibilita acessar arquivos externos, atualizar variáveis estatísticas, ou gerar gráficos, entre outras aplicações, o que dá uma grande flexibilidade de execução aos modelos simulados.

9.2.4. Atualização de Variáveis Estatísticas

Design/CPN possui variáveis estatísticas que permitem acumular e acessar estatísticas sobre os dados gerados durante a execução dos modelos. As variáveis estatísticas disponíveis incluem desvio padrão, média, entre outras.

É possível também dispor de gráficos gerados e atualizados automaticamente durante a simulação de um modelo. Esses gráficos podem também acessar variáveis estatísticas.

9.2.5. Modelagem Hierárquica

Design/CPN oferece dois meios para criar modelos hierárquicos: as **transições de substituição** e a **fusão de lugares**.

As transições de substituição são transições que podem conter em seu interior sub-redes distintas, o que torna possível a construção de modelos com vários níveis hierárquicos. Isso faz com que a modelagem em Design/CPN possa ser do tipo *top-down* ou *bottom-up*, ou uma combinação dos dois.

A fusão de lugares permite que sub-redes construídas em partes diferentes do modelo (por exemplo, em páginas diferentes) formem um só bloco, através de diversas instâncias do mesmo lugar.

9.3. Os Modelos Dinâmicos

Para avaliar as arquiteturas operacionais através de simulação, é necessário construir modelos em forma de RdP do comportamento dinâmico de cada componente. estes modelos genéricos (por exemplo, um modelo para os CLPs, outro para um tipo de protocolo de rede, etc.) devem ser ligados entre si de acordo com cada arquitetura analisada, e indicadores de desempenho devem ser aplicados ao modelo global do sistema.

Todos os modelos utilizados para simulação em Design/CPN estão apresentados no **ANEXO C**. No entanto, serão mostrados aqui alguns detalhes de construção dos modelos que levaram à obtenção de medidas de desempenho das duas arquiteturas operacionais propostas.

9.3.1. A Parte Operativa

A Figura 31 mostra o modelo da parte operativa do posto de montagem central no caso da Arquitetura B. Ao observar a estrutura deste modelo, nota-se a seqüência das diferentes operações da parte operativa: as transições *Chegada da Placa*, *Ler Codigo*, *Transporte*, *Montar Cubos* e *Manipular Placa*, *Montagem*, *Evacuar Placa* et *Gravar Codigo*.

A partir de *Ler Codigo*, pode-se observar que existem durações associadas às transições.

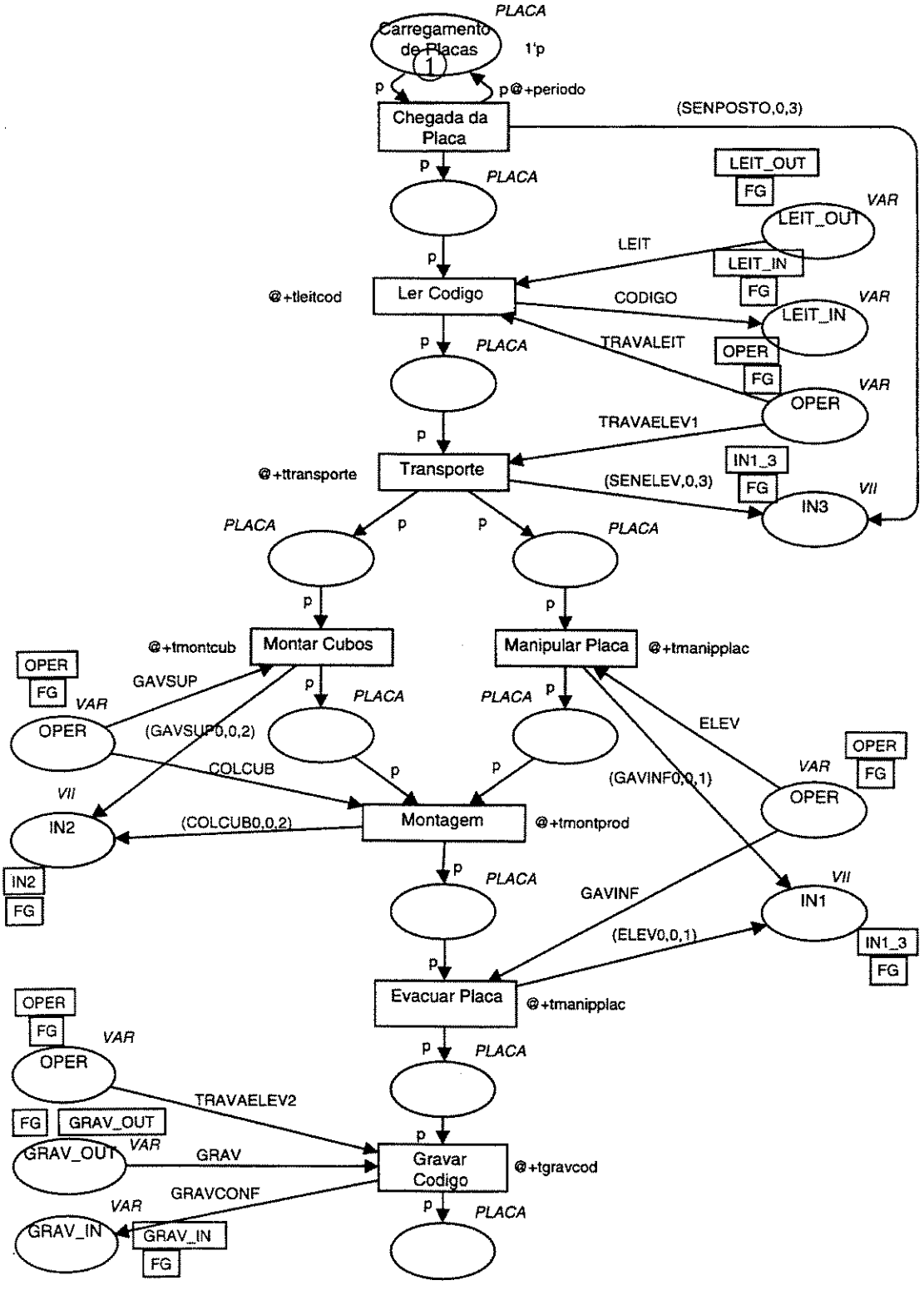


Figura 31. Rede de Petri modelando a parte operativa do posto de montagem central.

Por exemplo, a duração de *Ler Codigo* é indicada pela constante *tleitcod*, na declaração

@+tleitcod. Os valores dessas constantes são declarados no *quadro de declarações* (no ANEXO C) em que são declaradas todas constantes, variáveis, funções e cores do modelo. Ao lado de alguns lugares existem quadros com a forma FG. Isto indica o mecanismo de fusão de lugares, em que FG significa "Fusão Global". As diferentes fusões globais na Figura 31 são indicadas pelos quadros IN1_3, LEIT_OUT, LEIT_IN, OPER, IN2, GRAV_OUT et GRAV_IN. Por exemplo, todos lugares ao lado dos quais existe um quadro de fusão global IN1_3 (FG IN1_3) são na realidade o mesmo lugar (conseqüentemente contendo as mesmas marcas) de entrada dos equipamentos escravos 1 e 3. Assim, as fusões globais ligam lugares em diferentes páginas do modelo.

Esta figura mostra a marcação inicial do modelo, em que o número de marcas é mostrado através de um algarismo dentro de um pequeno círculo, ao lado do qual há uma expressão sublinhada, indicando os tipos (cores) individuais das marcas. Por exemplo, a marcação desta sub-rede consiste em uma marca *p*, de cor *PLACA*, no lugar *Chegada da Placa*.

Isto modela a provisão de placas, que chegam de acordo com um período expresso pela legenda *p @+periodo* sobre o arco de entrada deste lugar.

9.3.2. Os CLPs

Em geral, os controladores programáveis possuem sistemas de processamento **síncronos**. Isto significa que eles consideram suas entradas em instantes precisos, periódicos, e que eles executam o processamento dos dados no período seguinte [THO94]. Os sensores serão, então, periodicamente examinados e as saídas do sistema periodicamente atualizadas.

O modelo de um CLP, adaptado de [MEU97], é mostrado na Figura 32. Ele foi dividido em duas partes por uma linha tracejada. O lado esquerdo modela as fases do ciclo de varredura de um CLP: leitura, processamento e escrita dos sinais. As transições *fim Entrads*, *fim Process* et *fim Saídas* têm associadas a si durações (*tentr*, *tprocess1* et *tsaídas*) que determinam o tempo que uma marca *e* permanece em cada fase deste ciclo. O lado direito modela o fluxo dos dados, da entrada até a saída deste subsistema, passando pelo processamento lógico (modelado em outra página). Os arcos que cruzam a linha tracejada sincronizam o fluxo de dados com o ciclo do CLP.

Nota-se que o lugar de entrada do CLP (IN1, na parte superior à direita) faz parte da fusão global IN1_3. Portanto, ele é na verdade uma instância dos mesmos lugares vistos na página da parte operativa (Fig. 31).

Ainda na Figura 32, a saída desta sub-rede há a seleção dos diferentes dados de saída. As saídas de comunicação (x=COM12 orelse x=COM13) são dirigidas para a rede local enquanto as outras saídas (x<>COM12 andalso x<>COM13) são dirigidas para a acionar a parte operativa.

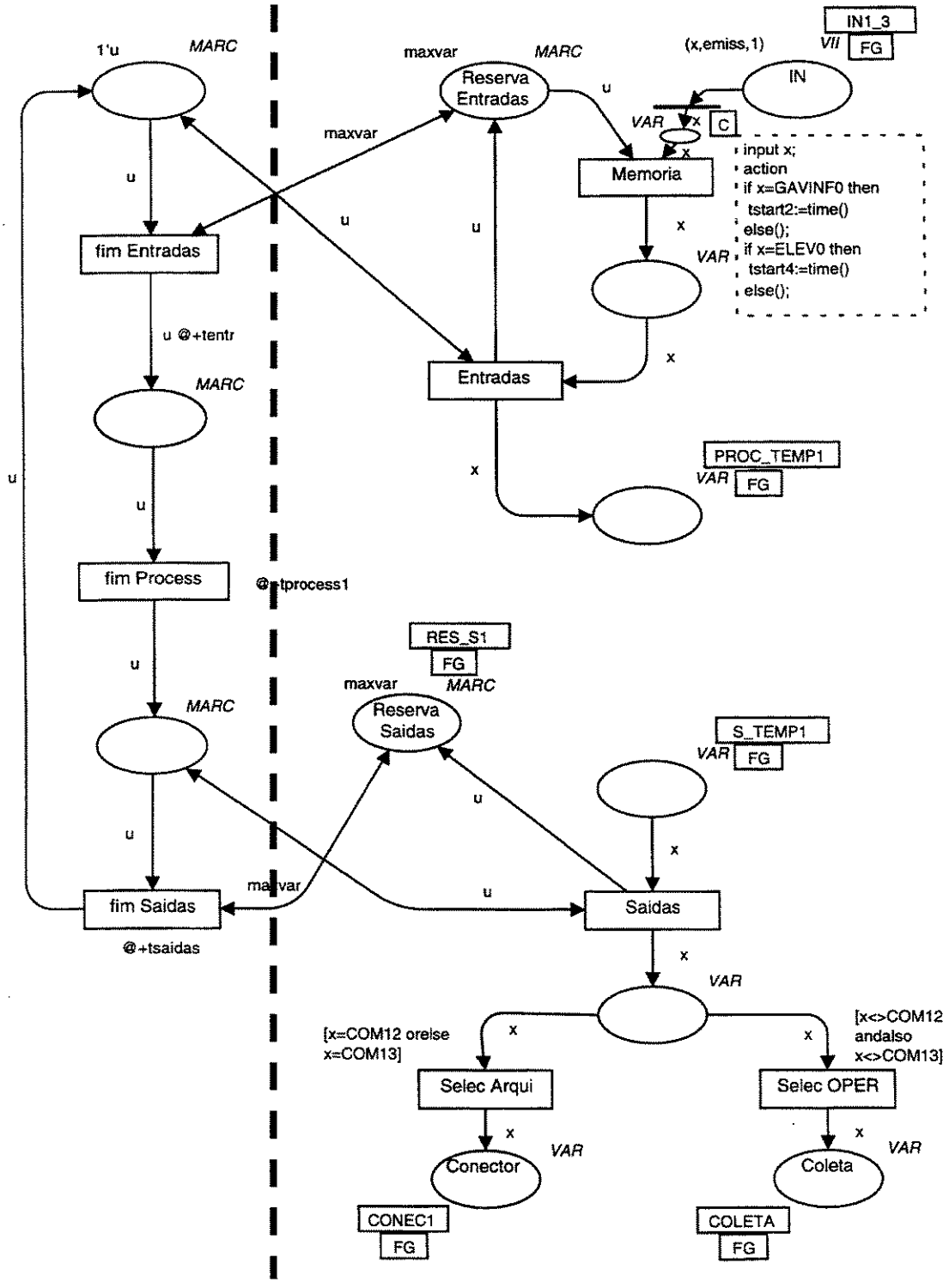


Figura 32. Modelo de CLP.

Finalmente, abaixo do lugar de entrada IN1 existe uma transição assinalada com um [C]. Isto indica que o disparo desta transição ativa a execução de código da Figura 33.

```
input x;
action
if x=GAVINF0 then
 tstart2:=time()
else();
if x=ELEV0 then
 tstart4:=time()
else()
```

Figura 33. Código para inicialização de variáveis.

O código é utilizado para inicializar e coletar as variáveis necessárias à obtenção dos indicadores de desempenho da arquitetura operacional. Neste caso, para medir o intervalo entre a detecção de GAVINF0 por parte do CLP1 e o acionamento de COLCUB por parte do CLP2 (ver os indicadores de desempenho na seção 9.1), deve-se armazenar em memória o instante em que GAVINF0 é recebido pelo CLP1. Isto é feito por meio da variável *tsatart2*. Da mesma forma, assim que esta transição é disparada por uma marca ELEV0, a variável *tstart4* é inicializada com o valor atual de tempo.

9.3.3. Processamento Lógico de Entradas e Saídas

Observa-se que esta sub-rede (Figura 34) é conectada àquela do CLP1 (Fig. 32) através das fusões globais RES_S1, PROC_TEMP1 e S_TEMP1. As declarações associadas à transição determinam as marcas de saída de acordo com as marcas de entrada. Por exemplo, se a transição é disparada por uma marca COM31, o arco de saída fará a transferência de uma marca ELEV para o lugar S_TEMP1. A modelagem deste processamento específico é relativamente simples, pois tratam-se de transformações lógicas de uma só entrada em uma só saída. Outros exemplos de processamento, mais complexos, estão apresentados no ANEXO C.

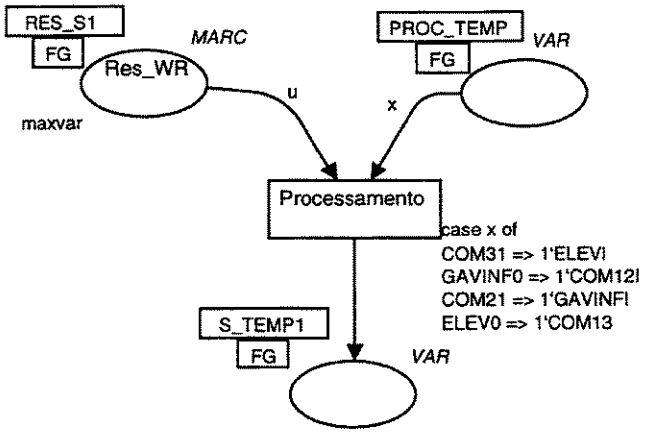


Figura 34. Processamento de dados no CLP 1.

9.3.4. A Rede Local

A RdP da Figure 35 modela uma rede de comunicação com protocolo do tipo mestre-escravo (do tipo *UNITELWAY* ou *Profibus-DP*) [MEU97]. Existem dois lugares de entrada na parte superior: o lugar à esquerda para os equipamentos escravos e o da direita para o mestre. No caso da arquitetura B, o equipamento 2 (CLP2) é o mestre na rede (fusão global OM2).

A escrutação periódica dos escravos é feita através da transição *Polling* e do lugar RPE. Este lugar contém a cada vez uma marca *emiss* de tipo número inteiro que determina o número do escravo que tem acesso à rede.

O disparo de *Polling* retira a marca *emiss* atual de RDE e deposita neste mesmo lugar uma nova marca *emiss* (modelando um novo número de escravo), por meio da função *prox(emiss)*. A função *prox(emiss)* (previamente declarada no *quadro de declarações*) faz com que *emiss* seja sucessivamente alternada entre os valores 1 e 3 (os números dos escravos). Após uma duração igual a *tpoll* (o período com que o *polling* ocorre), esta marca esta disponível para disparar mais uma vez a transição *Polling*.

A estrutura desta RdP dirige de forma diferente os dados destinados ao mestre e aqueles

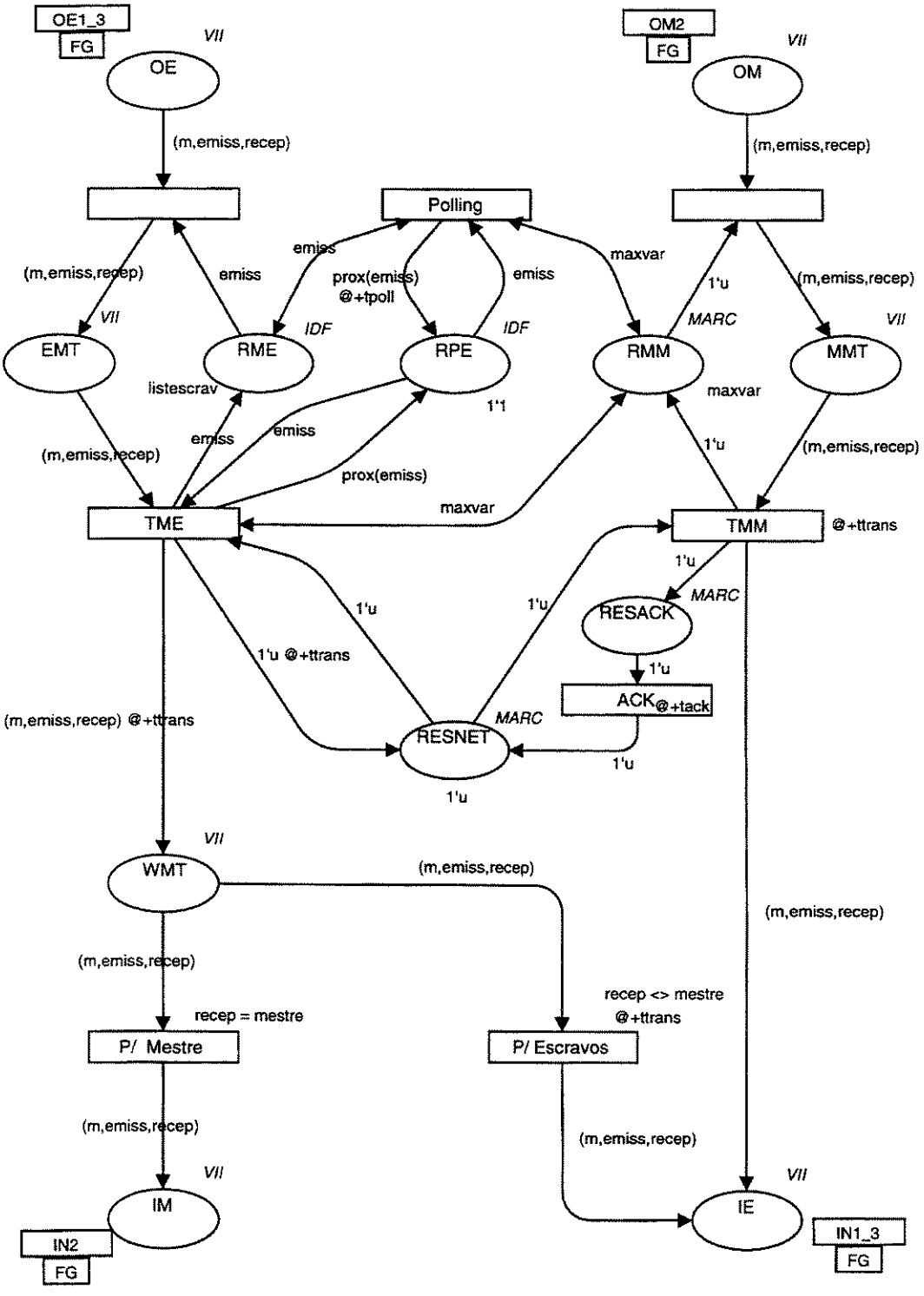


Figura 35. Rede de comunicação com protocolo do tipo mestre-escravo com polling cíclico [MEU97].

destinados aos escravos, levando em conta os tempos de transferência na rede e o tempo de

confirmação de recebimento (*acknoledgement*) entre os equipamentos (temporização @+*tack*, para as mensagens ACK). O comportamento de troca de mensagens é mostrado na Figura 36.

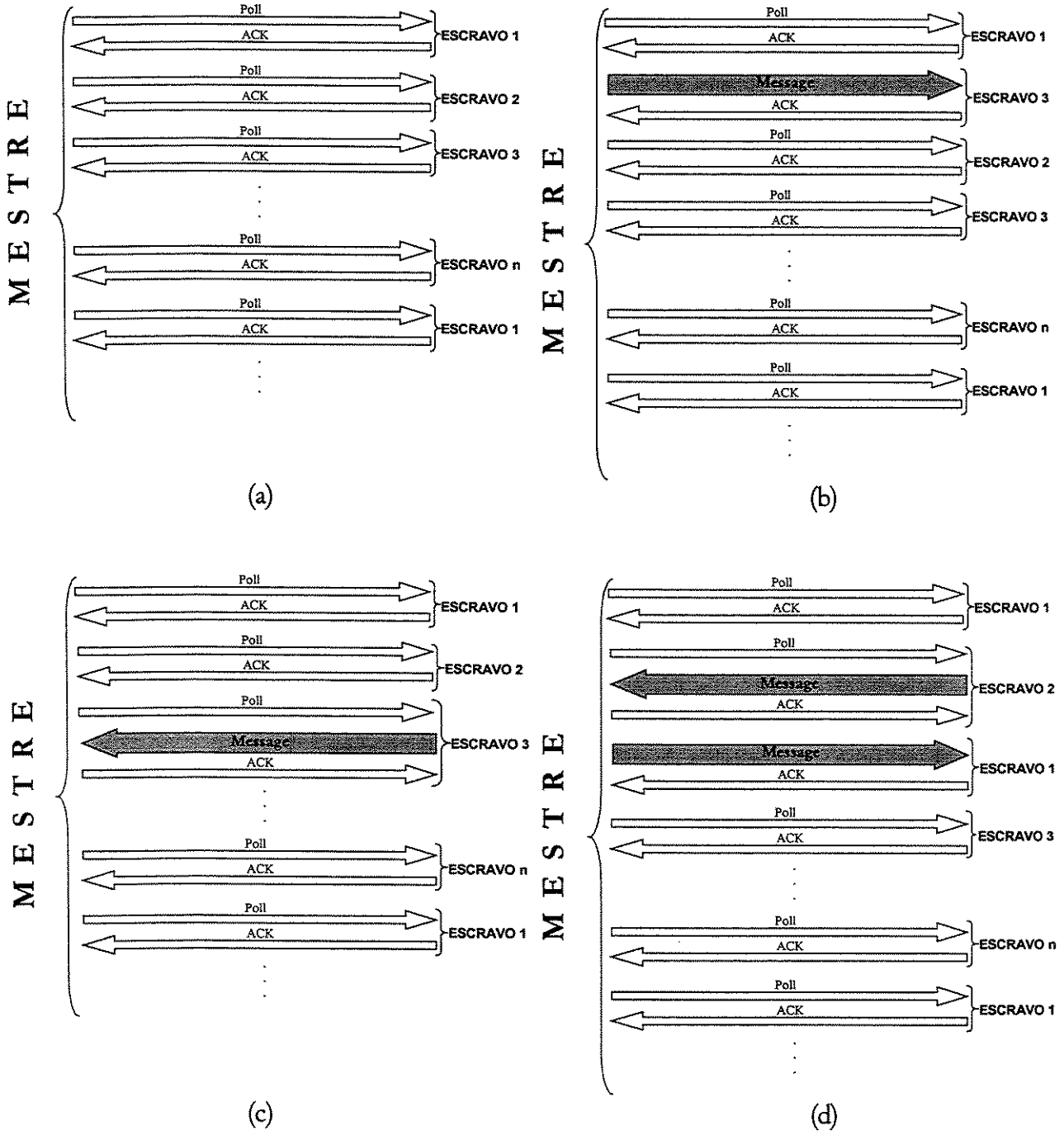


Figura 36. Mecanismo de escrutação cíclica (polling) dos escravos: (a) seqüência de polling dos escravos; (b) comunicação do mestre para um escravo; (c) comunicação de um escravo para o mestre; (d) comunicação de um escravo para outro.

9.3.5. A Placa Programável de Codificação Magnética

A RdP da Figura 37 modela o comportamento da placa programável. A placa *BALOGH®* não apresenta um sistema de processamento síncrono como os controladores programáveis. Ela segue uma seqüência fixa de linhas de instruções que determinam fases bem definidas e não cíclicas de operações como cálculo, tratamento lógico, leitura e escrita. Portanto, pode-se considerar que a placa tem um comportamento determinístico do ponto de vista de tempos de operação.

Esta RdP dirige os dados para diferentes equipamentos ligados à placa: os sensores e atuadores do sistema de transferência (SENPOSTO, SENELEV, TRAVALEIT e TRAVAELEV para a parte operativa); os comandos e requisições para os módulos de diálogo (LEIT, GRAV, REQLEIT e REQGRAV para o barramento CAN); e as mensagens de comunicação para os CLPs (COM31 e COM32 para a rede local).

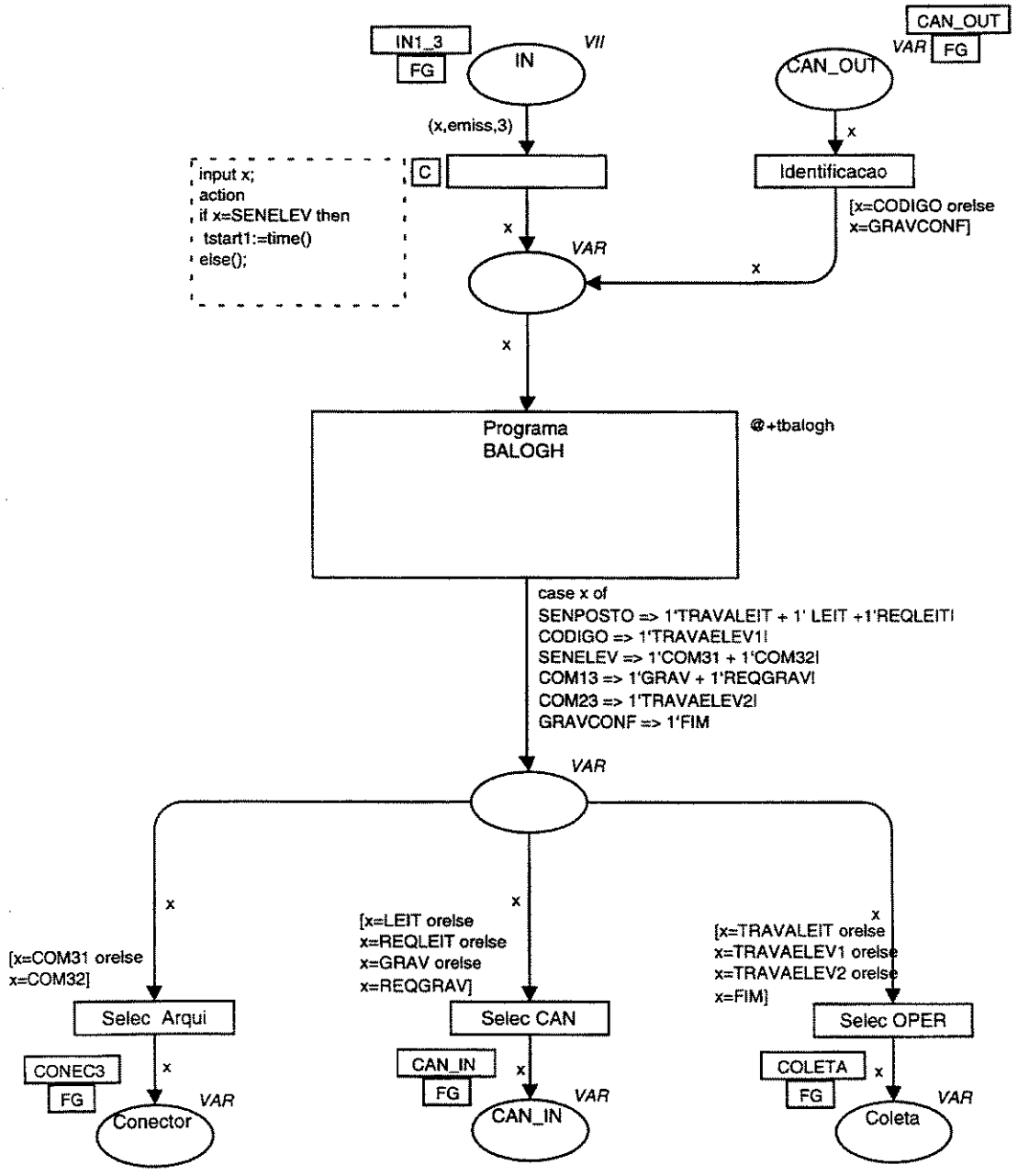


Figura 37. Modelo da placa BALOGH®.

9.3.6. O Barramento CAN

Segundo o protocolo CAN (Controler Area Network), todos os nódulos ligados à rede podem emitir mensagens simultaneamente. Se há colisões, o acesso ao barramento bus é "ganho" por meio da presença de um bit dominante (*bit* 0) no início de uma mensagem.

Uma mensagem. emitida por um equipamento é recebida por todos os outros através do mecanismo de difusão (*broadcast diffusion*). A primeira seqüência de bits de cada mensagem é uma identificação, através da qual um equipamento reconhece uma mensagem destinada a ele, ao que se denomina *filtragem*.

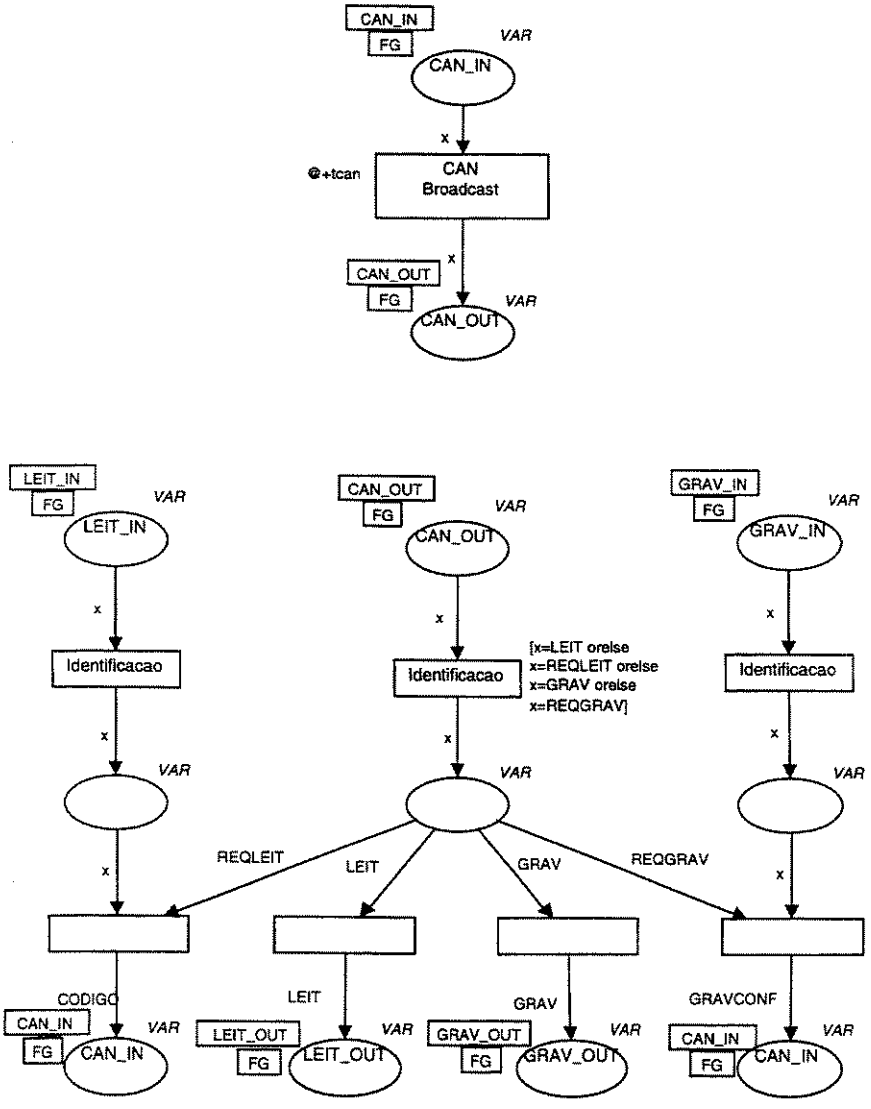


Figura 38. O barramento CAN.

9.4. Resultados

As figuras seguintes foram geradas durante simulação por Design/CPN a partir das linhas de código adicionadas à estrutura das RdP. Mais precisamente, a geração dos gráficos foi feita a partir das sub-redes *Coleta* (ANEXOS C.1.9 e C.2.10).

9.4.1. Comunicação do Equipamento 3 para os Equipamentos 1 e 2

(Figs. 39 e 40) Como na Arquitetura A trata-se de uma mensagem do mestre para dois escravos, a única diferença entre a ativação de *Elevador* e *Gaveta Superior* é a ordem aleatória de emissão das mensagens pelo mestre. Algumas vezes a emissão de COM31 acontece antes da emissão de COM32, às vezes o contrário, com um intervalo médio de 100 ms entre ativação dos dois atuadores.

Como na Arquitetura B o CLP 2 é o mestre, ele sempre tem a prioridade de comunicação na rede e, portanto, um acesso mais rápido à informação vinda da placa programável. Por sua vez, a transferência de uma mensagem do escravo 3 (placa) para o escravo 1 (CLP 1) é retardada pela hierarquia de acesso à rede. Mesmo quando um caso singular, em que todas as condições são desfavoráveis ao CLP 2 e favoráveis ao CLP 1 (por exemplo, se uma mensagem chega assim que o CLP 2 passou à fase de processamento, tendo que esperar por um ciclo completo para ler suas entradas), o intervalo entre SENELEV (presença placa elevador) e ELEV (Elevador) não é tão inferior ao intervalo entre SENELEV e GAVSUP (Gaveta superior).

Um outro dado importante é a defasagem temporal entre o acionamento dos dois atuadores. Percebe-se pelos resultados obtidos que, se na Arquitetura B esta defasagem apresenta um mínimo de poucos milissegundos, seu máximo é maior que o valor médio obtido na Arquitetura A (a defasagem chega a picos de 200 ms na Arquitetura B enquanto esta se concentra por volta de 100 ms na Arquiteura A). Por isso, se um dos pré-requisitos do sistema fosse que a defasagem entre o acionamento dos dois atuadores não pudesse ultrapassar um determinado valor limite (150 ms, por exemplo), a escolha de arquitetura recairia sobre a opção A.

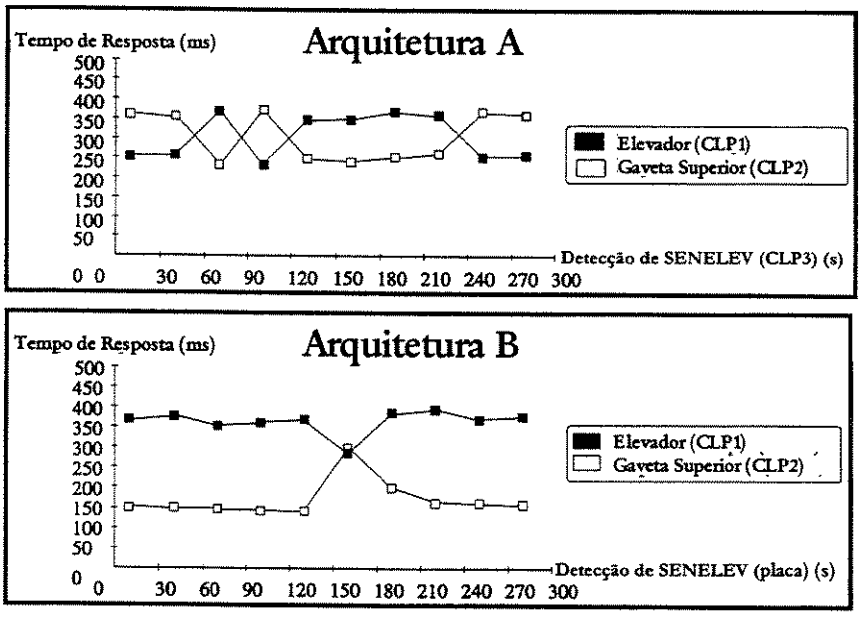


Figura 39. Intervalo entre presença placa elevador e Elevador+Gaveta Superior.

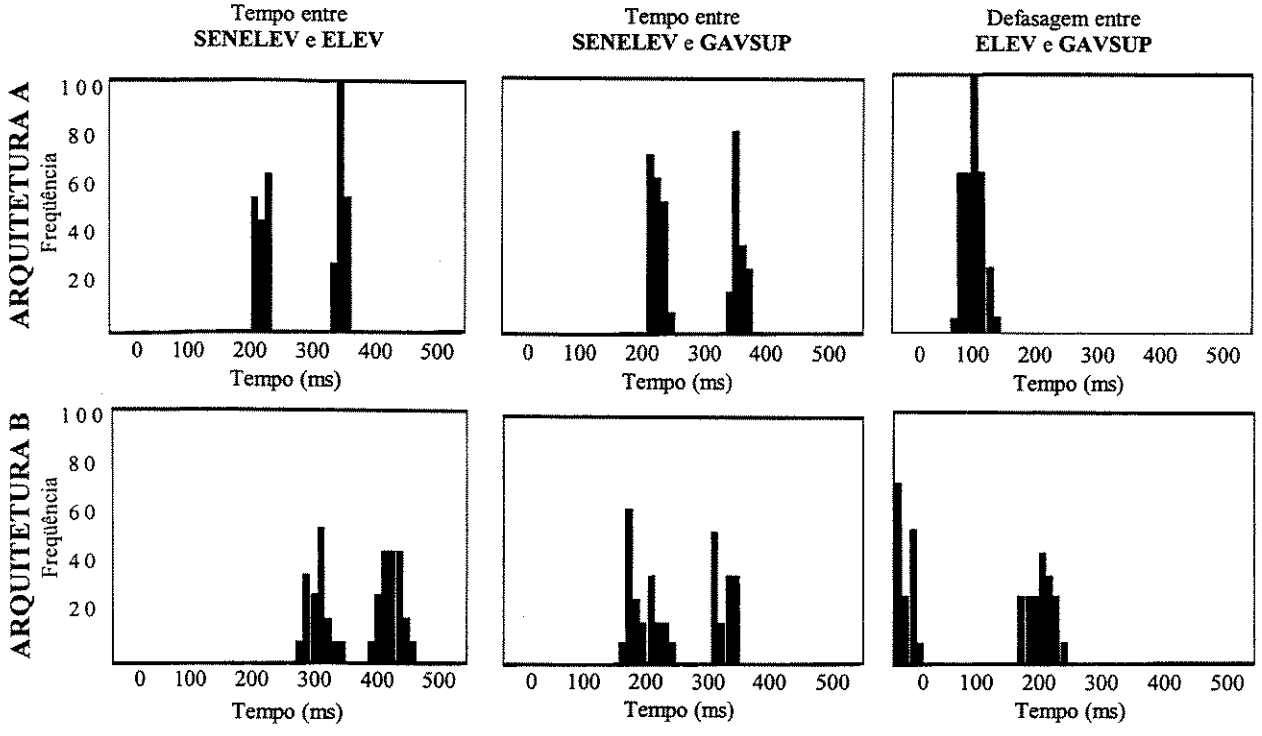


Figura 40. Resultados de SENELEV ==> ELEV+GAVSUP.

9.4.2. Comunicação do Equipamento 1 para o Equipamento 2

A diferença de resultados visível nas Figuras 41 e 42 é causada pela diferença de prioridade entre uma comunicação escravo/ escravo (Arquitetura A) e uma comunicação escravo/mestre da rede (Arquitetura B).

No caso da Arquitetura A, como tanto o emissor como o receptor são escravos, cada um deve esperar sua vez para emitir ou para receber uma mensagem. A Arquitetura B fornece melhores tempos de reação pois mesmo se o emissor deve esperar por sua vez, a mensagem é enviada diretamente ao mestre da rede.

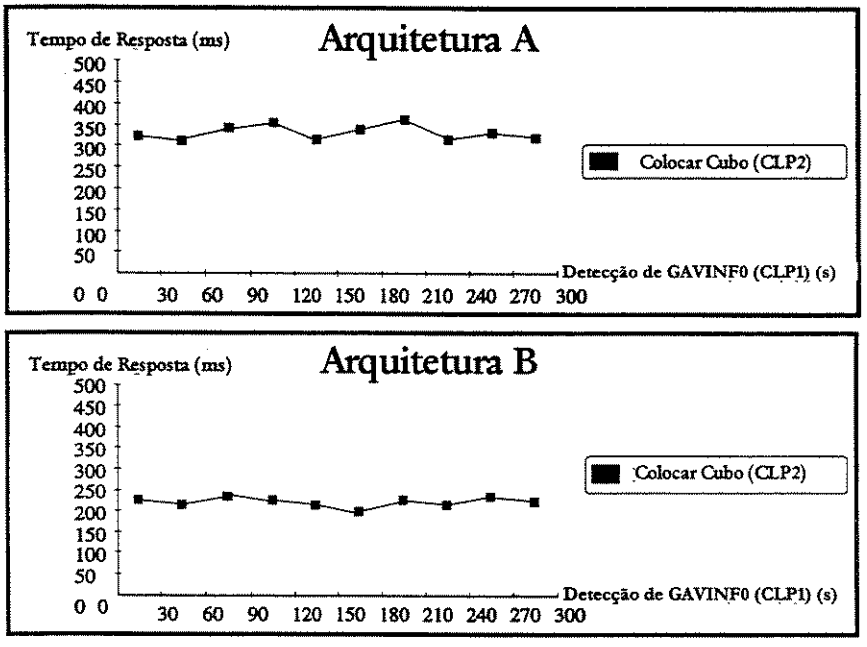


Figure 41. Intervalo entre gaveta inferior IN e Colocar Cubo.

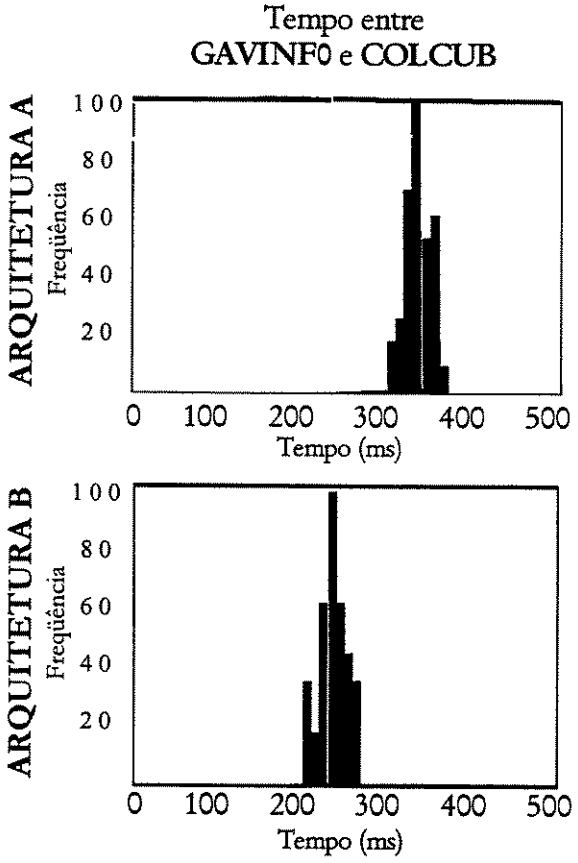


Figure 42. Resultados de gaveta inferior IN ==> Colocar Cubo.

9.4.3. Comunicação do Equipamento 2 para o Equipamento 1

(Figs. 43 e 44) As defasagens na Arquitetura A são maiores que na Arquitetura B, porque trata-se de comunicação escravo/escravo. No entanto, como a Arquitetura B exige que o CLP 2 emita paralelamente mensagens para o CLP 1 e para a placa (equipamento 3) (ver o GRAFCET do ANEXO B.2.2), dependendo da ordem das emissões, o tempo de ativação de *Trava Placa Elevador* é um pouco comprometida, mas sempre inferior ao obtido na Arquitetura A.

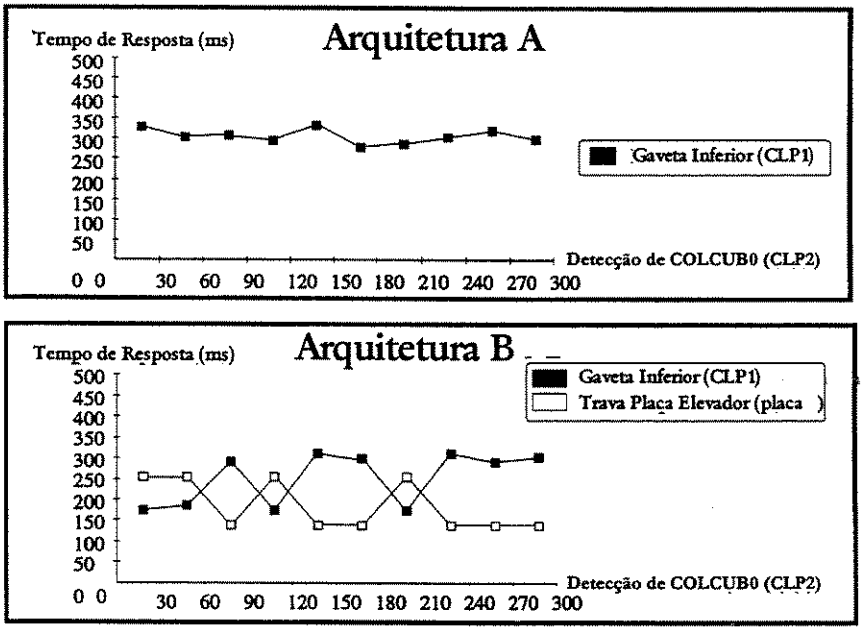


Figura 43. Intervalo entre colocar cubo IN e Gaveta Inferior.

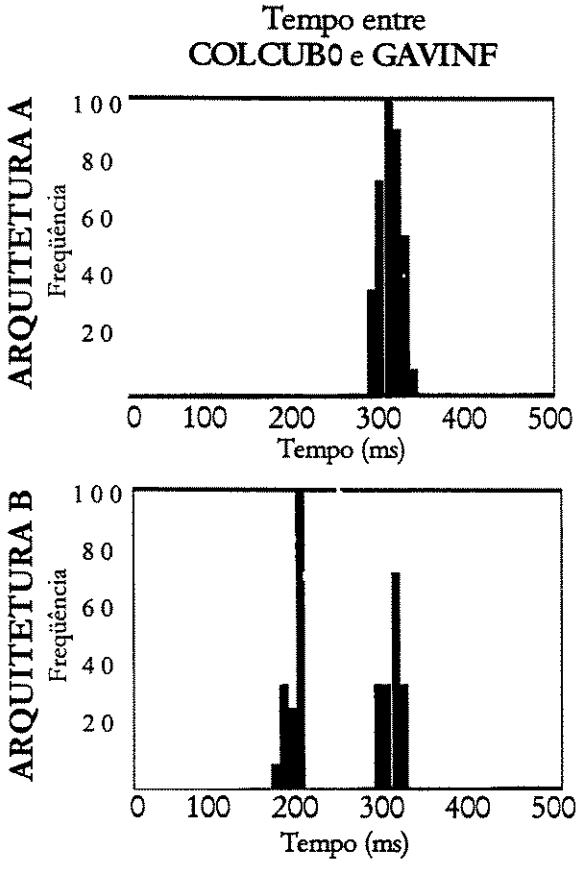


Figura 44. Resultados de colocar cubo IN ==> Gaveta Inferior.

9.4.4. Comunicação do Equipamento 1 para o Equipamento 3

(Figs. 45 e 46) Durante a modelagem da dinâmica das entrada e saídas do comando, adicionou-se uma marca de contagem de placas montadas, chamada *FIM*. Esta marca é o resultado do último tratamento de dados da função Processar Código, e sinaliza o fim de um ciclo de montagem, após a placa ter sido levada por *Elevador* até a esteira do sistema de transferência (ELEV0 – *elevador IN*). Os gráficos seguintes mostram os intervalos entre a detecção de ELEV0 por parte do CLP 1 e a geração de uma possível saída (modelando, por exemplo, a atualização de um contador no console do operador da plataforma) por parte do equipamento 3.

Como o equipamento 3 (CLP 3) é o mestre da rede na Arquitetura A, os tempos de comunicação para o CLP 1 são melhores que na Arquitetura B.

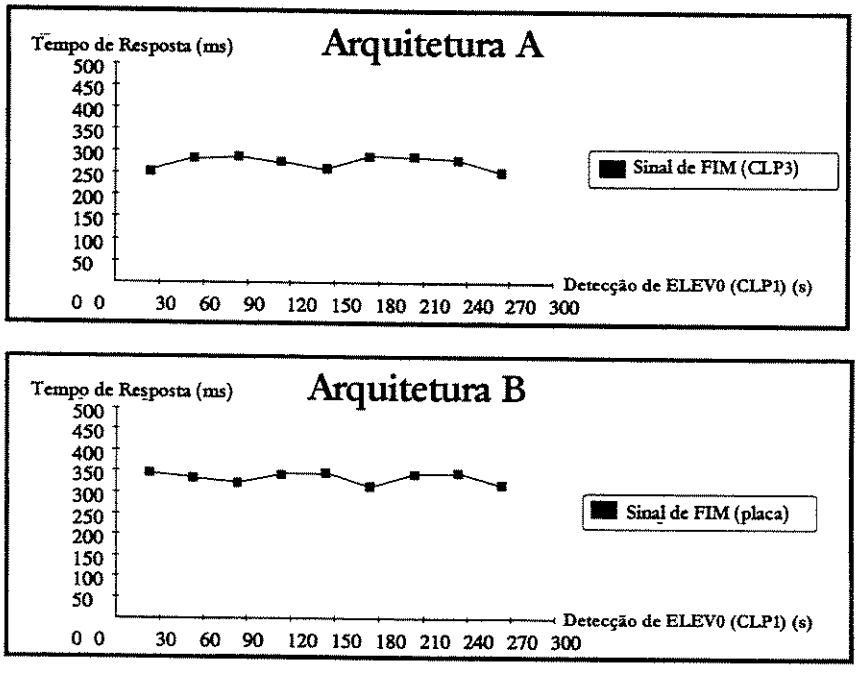


Figura 45. Intervalo entre elevador IN e o fim de operação de uma placa.

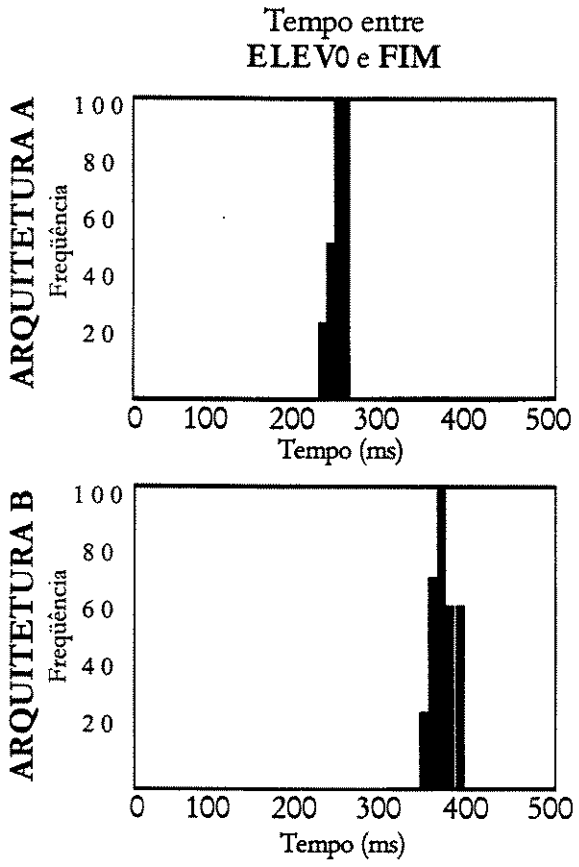


Figura 46. Resultados de ELEV0 ==> FIM.

9.5. Avaliação de desempenho baseada nos resultados

A escolha de uma ou outra solução de arquitetura de comando depende de quais indicadores de desempenho são críticos para a aplicação do SAP.

Por exemplo, retomando os primeiros resultados, os da comunicação do equipamento 3 para os equipamentos 1 e 2 (seção 9.4.1), nota-se que:

- se o tempo de acionamento do atuador Elevador é determinante (tiver que ser mínimo), a Arquitetura A seria a melhor escolha;
- se o tempo de acionamento do atuador Gaveta Superior é determinante (tiver que ser mínimo), escolher-se-ia a Arquitetura B;

- se a defasagem entre os dois acionamentos tiver que ser inferior a um patamar de 150 ms, por exemplo, a Arquitetura A seria a opção acertada.

No caso da comunicação do equipamento 1 para o equipamento 3 (seção 9.4.4), se o tempo de disparo da mensagem de fim de operação devesse ser a menor possível, escolher-se-ia então a Arquitetura A.

Assim, percebe-se que são os pré-requisitos da aplicação em questão que indicam à equipe de projeto como utilizar os dados extraídos desta metodologia. A partir da comparação entre os resultados obtidos e os pré-requisitos do SAP, consegue-se estabelecer os critérios para a escolha da melhor solução de arquitetura de comando.

Capítulo 10

Conclusões e Perspectivas

Este trabalho apresentou os elementos necessários à modelagem das propriedades da arquitetura de comando de um sistema distribuído.

De um lado, é preciso dispor de uma representação das partes do sistema de comando (lógicas e materiais) e das relações entre essas partes e o ambiente com o qual interage este sistema. As ferramentas provenientes da Análise Estruturada servem ao propósito de construir de forma inteligível e sistemática um modelo estático da estrutura de um sistema complexo.

Por outro lado, é necessário dispor também de um meio formal capaz de fornecer os mecanismos para representar dinamicamente o comportamento desta estrutura. Desta forma, é possível simular o modelo do sistema. O paradigma de Redes de Petri coloridas e temporizadas é bem adaptado à simulação do comportamento de sistemas complexos, tais como sistemas com sincronização ou com um grau não desprezível de paralelismo.

Foi necessário reunir estas duas abordagens para obter um modelo suficientemente completo e para avaliá-lo em termos de propriedades importantes para a equipe de projeto, tais como o desempenho temporal do sistema.

Aplicado a um estudo de caso, o processo de modelagem, desde a utilização da Análise estruturada até a modelagem dinâmica através de um editor e simulador de Redes de Petri

coloridas e temporizadas mostrou-se ser um meio satisfatório para uma avaliação preliminar das arquiteturas operacionais de um sistema distribuído.

A modularidade e genericidade dos modelos sob forma de RdP coloridas abre a possibilidade da construção de bibliotecas de modelos. Modelos para tipos de equipametos diferentes, ou de áreas tecnológicas específicas. Desta forma, os projetistas teriam à disposição ferramentas de pronto uso no trabalho de projeto de uma arquitetura de comando.

As perspectivas de continuação deste trabalho é a criação de cenários mais complexos, utilizando Redes de Petri estocásticas, para que este processo de modelagem seja mais adaptado às variações de desempenho intrínsecas à transmissão de sinais (casos de bloqueio de um receptor, mensagens de tamanhos variáveis, entre outros aspectos) cujo comportamento está longe de ser determinístico.

REFERÊNCIAS BIBLIOGRÁFICAS

[BOT93] BOTTI O., DeCINDIO F., *Process and ressource boxes: an integrated PN performance model for application and architectures,* IEEE International Conference on Systems, Man and Cybernetics, 1993.

[BRA83] BRAMS G. W., *Réseaux de Petri: théorie et pratique*, tomos 1 et 2, Masson Editions, 1983.

[CHE76] CHEN P., *The entity-relationship model – toward a unified view of data*, ACM Transaction on Database Systems, vol. 1, n° 1, março 1976.

[COU97] COUFFIN F., *Modèle de données de reférence et processus de spécialisation pour l'intégration des activités de conception en génie automatique*, tese de doutorado da Ecole Normale Supérieure de Cachan, 1997.

[DeM79] DeMARCO T., *Structured analysis and system specification*, Yourdon Press, Prentice-Hall, Englewood Cliffs, 1979.

[DEN94] DENIS B., *Assistance à la conception et à l'évaluation de l'architecture de conduite des systèmes de production complexes*, relatório de tese, Université Nancy I, 1994.

[DES93a] *DESIGN/CPN Tutorial for X-Windows*, Meta Software Corporation, 125, Cambridge Park Drive, Cambridge MA, 02140, USA.

[DES93b] *DESIGN/CPN Reference Manual*, Meta Software Corporation, 125, Cambridge Park Drive, Cambridge MA, 02140, USA.

[EDW93] EDWARDS K., *Real-time structured methods: systems analysis*, John Wiley & Sons, 1993.

[FDI89] FDIDA S., PUJOLLE G., *Modèles de systèmes et de réseaux – tome 1: performance*, Editions Eyrolles, Paris, 1989.

[HAT91] HATLEY D. J., PIRBHAI I. A., *Stratégies de spécification des systèmes temps réel (SA-RT)*, tradução do original americano *Strategies for real-time system specification*, Masson (Méthodes Informatiques et Pratique des Systèmes), Paris, 1991.

[HEU90] HEUSER C.A., *Modelagem conceitual de sistemas – Redes de Petri*, V Escola Brasileiro-Argentina de Informática (EBAI), Nova Friburgo, 1990.

[ICAM80] *Architect's manual, Integrated Computer Aided Manufacturing (ICAM) definition method IDEF-0*, documento editado pela USAF, ref DR-80-ATPC-01, abril 1980.

[IEC88] International Electrotechnical Comission, *Etablissement des diagrammes fonctionnels pour des systèmes de commande*, norma internacional IEC 848, 1988.

[JAU90] JAULENT P., *Génie logiciel: les méthodes (SADT, SA, E-A, SA-RT, SREM, SYS-P-O, OOD, HOOD...)*, Armand Colin, 1990.

[JEN87] JENSEN K., *Coloured Petri Nets*, em *Petri Nets: central models and their properties*, Lecture Notes in Computer Science, 254, Springer-Verlag, Berlin, pp. 248-299, 1987.

[MEU97] MEUNIER P., DENIS B., *Validation du comportement dynamique des architectures de conduite des systèmes de production par simulation*, Modélisation et simulation des systèmes de production et logistique, pp. 229-238, Rouen, France, junho 1997.

[MUR95] MURATA T., *Application of Petri Nets to Sequence Control Programming*, em *Petri Nets in Flexible and Agile Automation*, ZHOU M. C. (editor), Kluwer Academic Publishers, Boston, 1995.

[NEE87] NEELAMKAVIL F., *Computer simulation and modeling*, John Wiley & Sons, 1987.

[PAR96] PARET D., *Le bus CAN,* Dunod, Paris, 1996.

[PRI86] PRITSKER A. A. B., *Introduction to simulation and SLAM II*, John Wyley & Sons, 1986.

[REI82] REISIG W., *Petri nets: an introduction*, Springer-Verlag, 1982.

[ROB94] ROBERTAZZI T. G., *Computer networks and systems: queueing theory and performance evaluation*, segunda edição, Springer-Verlag, 1994.

[SAL96] SALAÜN P., Modélisation des caractéristiques temporelles des systèmes numériques de contrôle-commande (SNCC), Modélisation des Systèmes Reactifs, AFCET, pp. 27-33, Brest, França, março 1996.

[THO94] THOMESSE J.-P., *Réseaux locaux industriels. Typologie et cacactéristiques*, Techniques de l'Ingénieur, tratado Mesures et Contrôle, documento R 7 576, 1994.

[TOU91] TOUTAIN L., *SAMSON: un simulateur pour systèmes répartis et temps-réel*, tese de doutorado da Université du Hauvre, 1991.

[VEN95] VENKATESH K., ZHOU M. C., CAUDILL R. J., *Discrete Event Control Design for Manufacturing Systems Via Ladder Logic Diagrams and Petri Nets: A Comparative Study*, em *Petri Nets in Flexible and Agile Automation*, ZHOU M. C. (editor), Kluwer Academic Publishers, Boston, 1995.

[VER91] VERDIN L., *De la spécification à l'exploitation: le cycle de vie des SAP*, Actes du GAMI: Génie Automatique et Production Industrielle, vol. 2, Saint-Ouen, France, março 1991.

[VIA99] VIANA L. C., ROSÁRIO J. M., FAURE J. M., *Performance evaluation of control architectures in industrial automation*, 15th International Conference on CAD/CAM Robotics & Factories of the Future (CARSFOF), Águas de Lindóia, agosto 1999.

[WAR86] WARD P. T., MELLOR S. J., *Structured development for real-time systems*, vol. 3, Englewood Cliffs, Prentice-Hall (Yourdon Press computing series), 1986.

[YOU89] YOURDON E., *Modern structured analysis*, Yourdon Press, Prentice Hall, Englewood Cliffs, NJ, 1989.

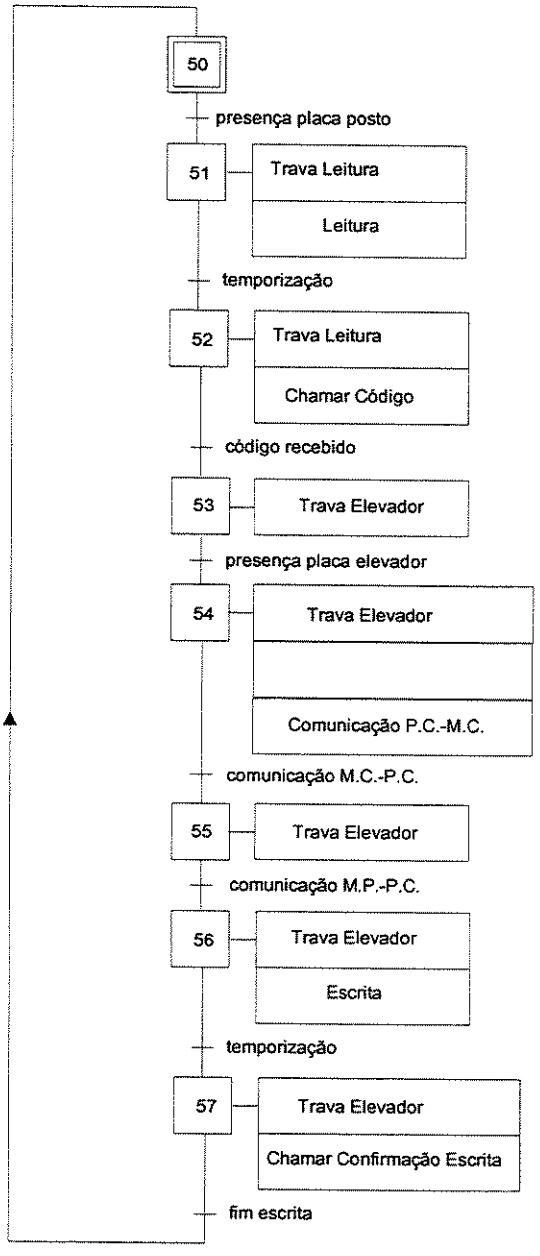
[ZAY97] ZAYTON J., LESAGE J. J., DÉPLANCHE A. M., *Vérification et validation du GRAFCET*, Journal Européen des Systèmes Automatisés, vol. 31, n° 4, pp. 713-740, Éditions Hèrmes, Paris, junho 1997.

[ZHO95] ZHOU M. C., ZURAWSKI R., *Introduction to Petri Nets in Flexible and Agile Automation*, em *Petri Nets in Flexible and Agile Automation*, ZHOU M. C. (editor), Kluwer Academic Publishers, Boston, 1995.

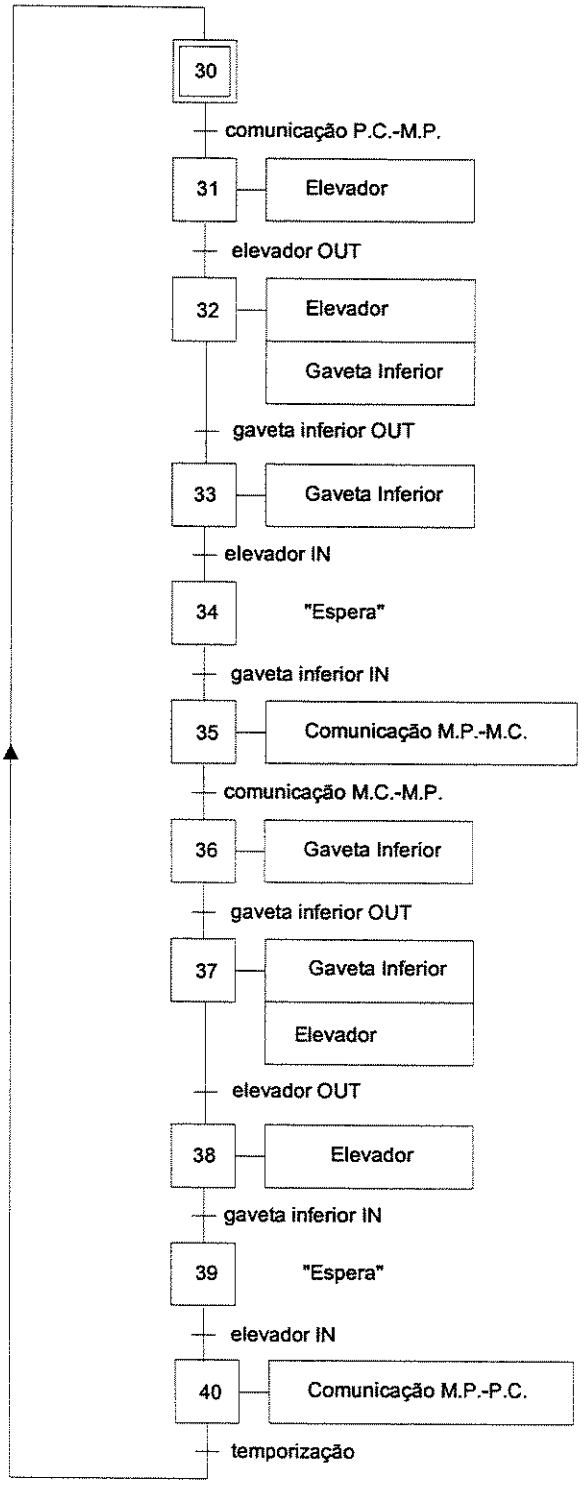
ANEXOS

ANEXO A: ARQUITETURA FUNCIONAL (MODELOS COMPORTAMENTAIS)

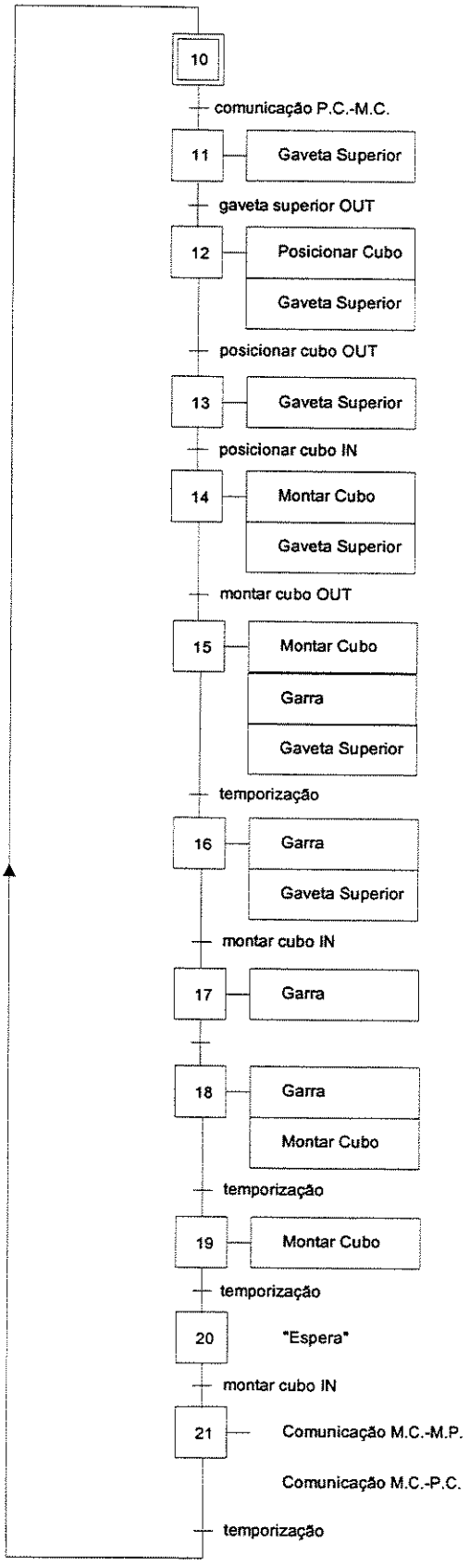
A.1. GRAFCET DA FUNÇÃO PROCESSAR CÓDIGO



A.2. GRAFCET DA FUNÇÃO MANIPULAR PLACA



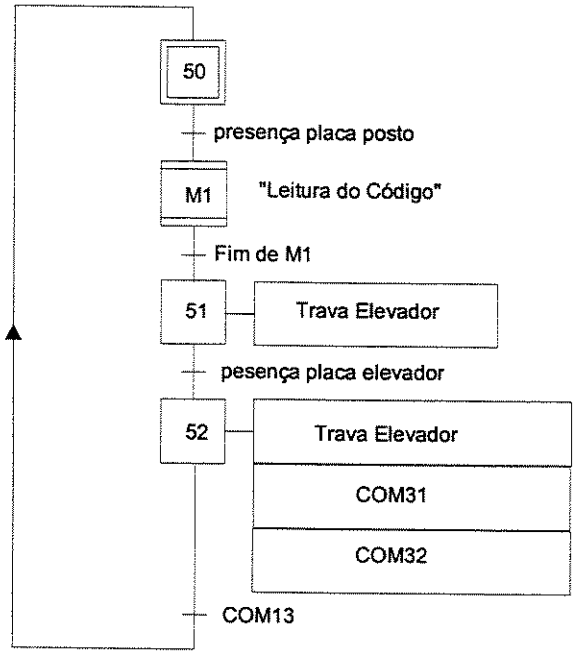
A.3. GRAFCET DA FUNÇÃO MONTAR CUBOS



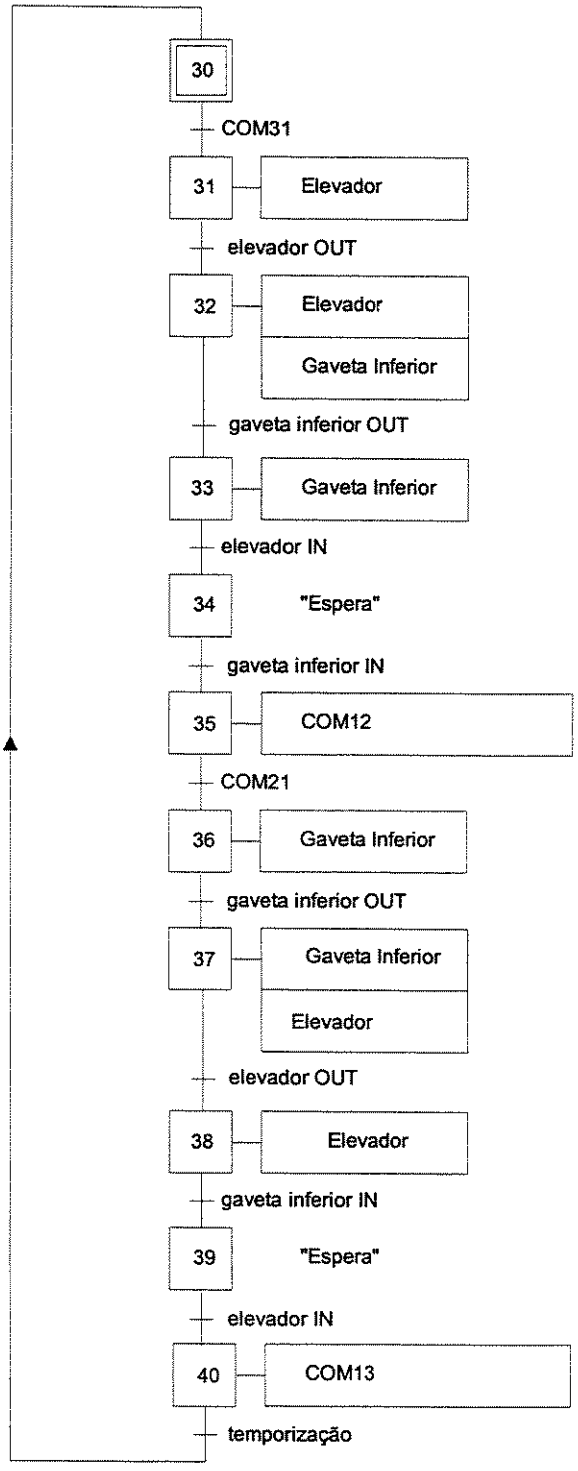
ANEXO B: ARQUITETURAS OPERACIONAIS

B.1. ARQUITETURA A

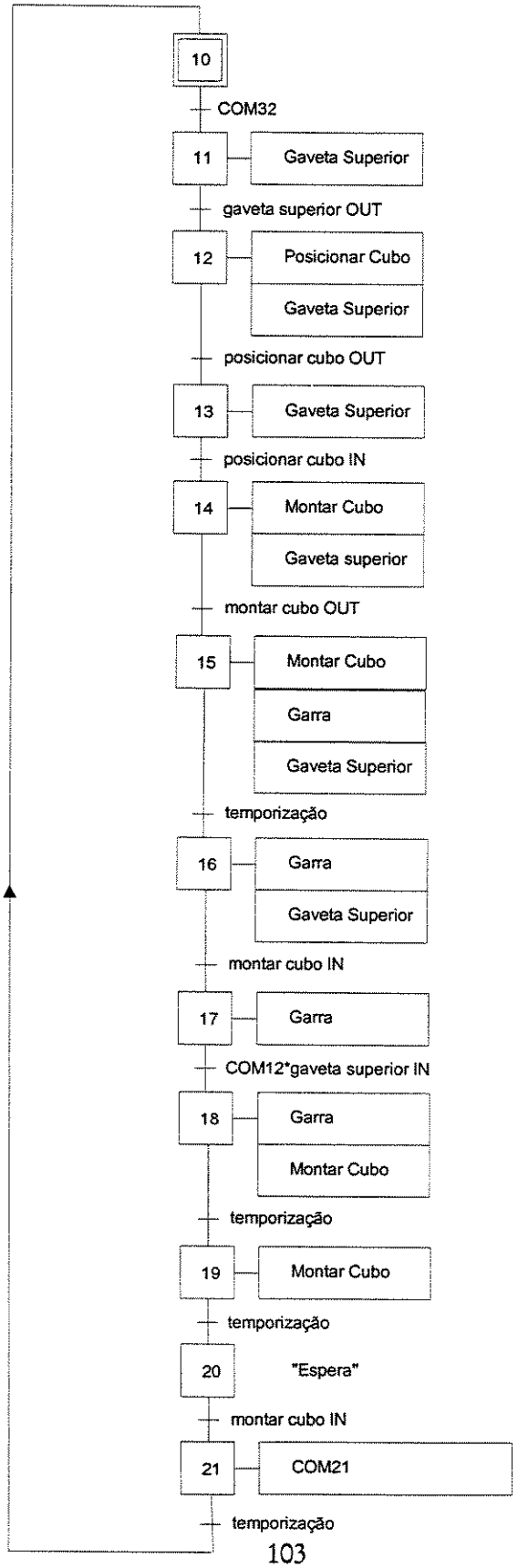
B.1.1. PROCESSAR CODIGO A



B.1.2. MANIPULAR PLACA A

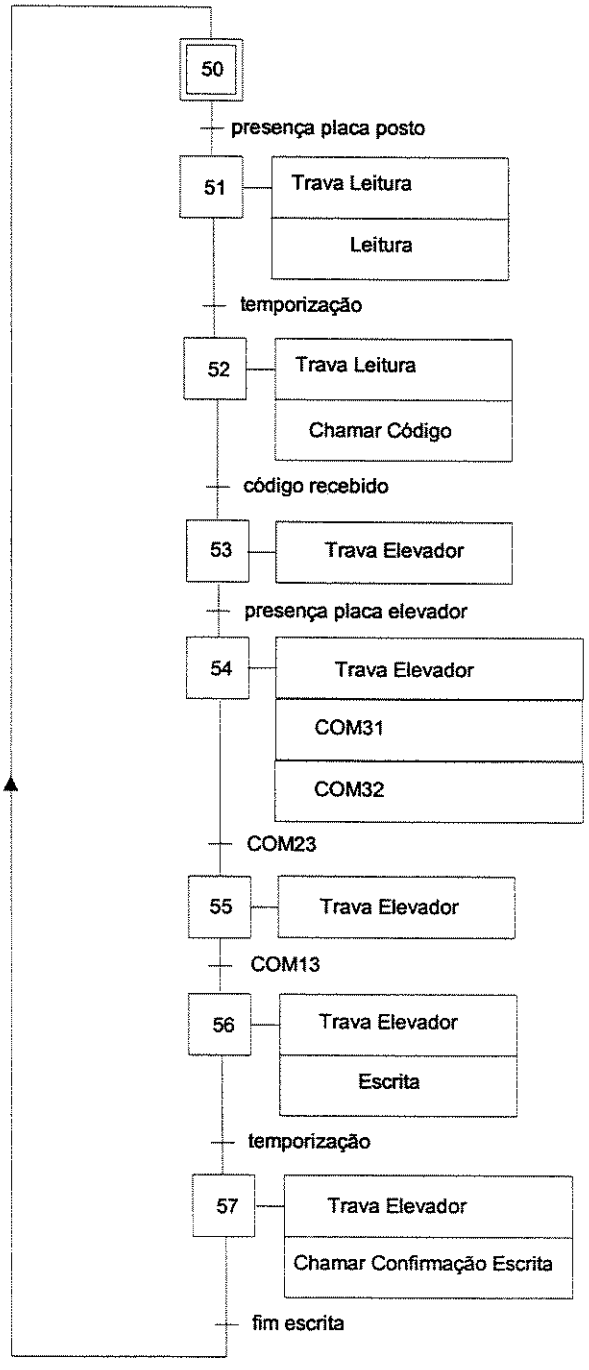


B.1.3. MONTAR CUBOS A

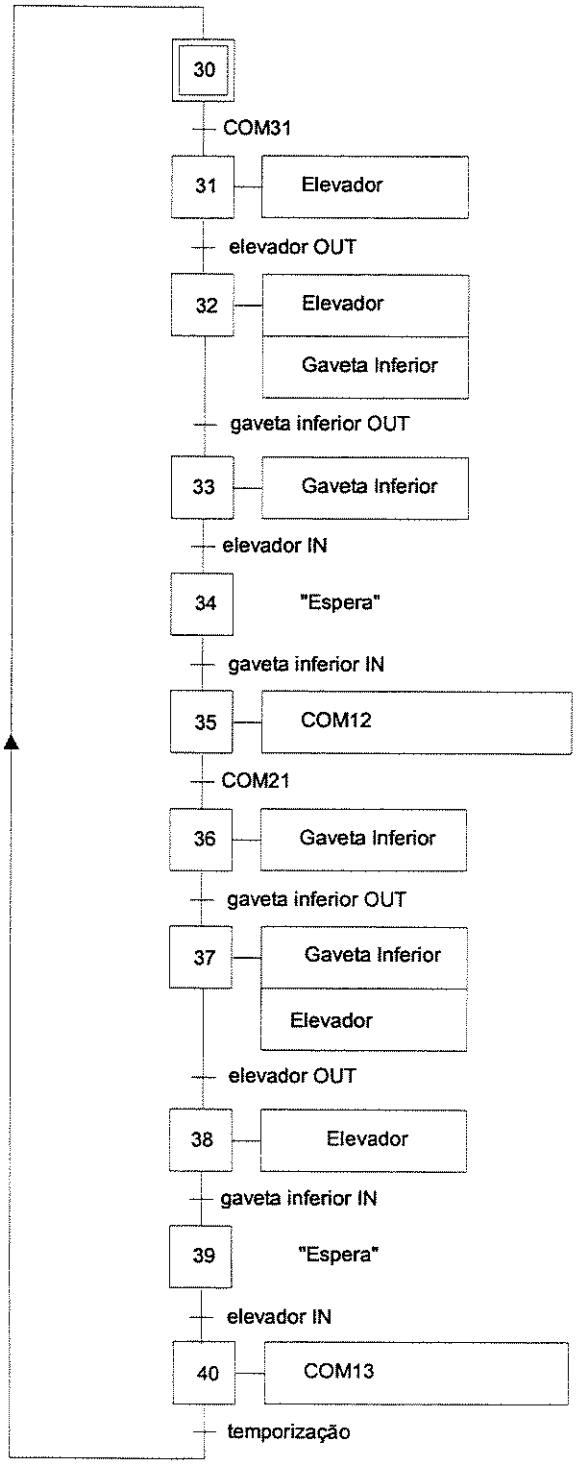


B.2. ARCHITECTURE B

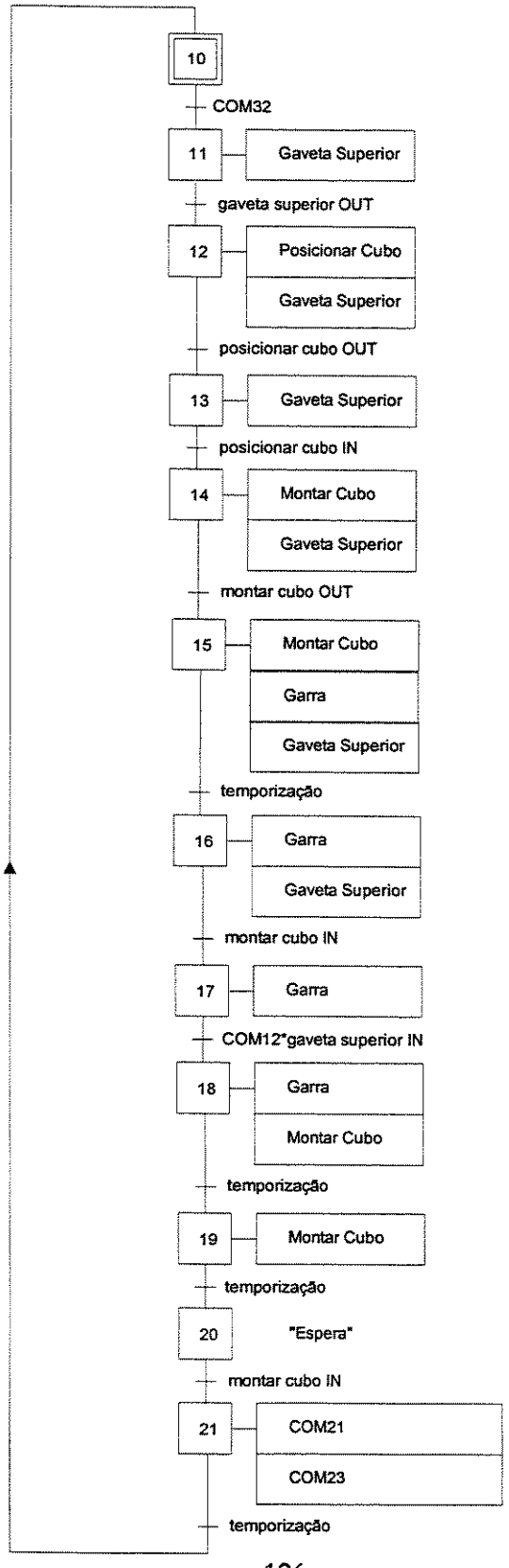
B.2.1. PROCESSAR CODIGO B



B.2.2. MANIPULAR PLACA B



B.2.3. MONTAR CUBOS B



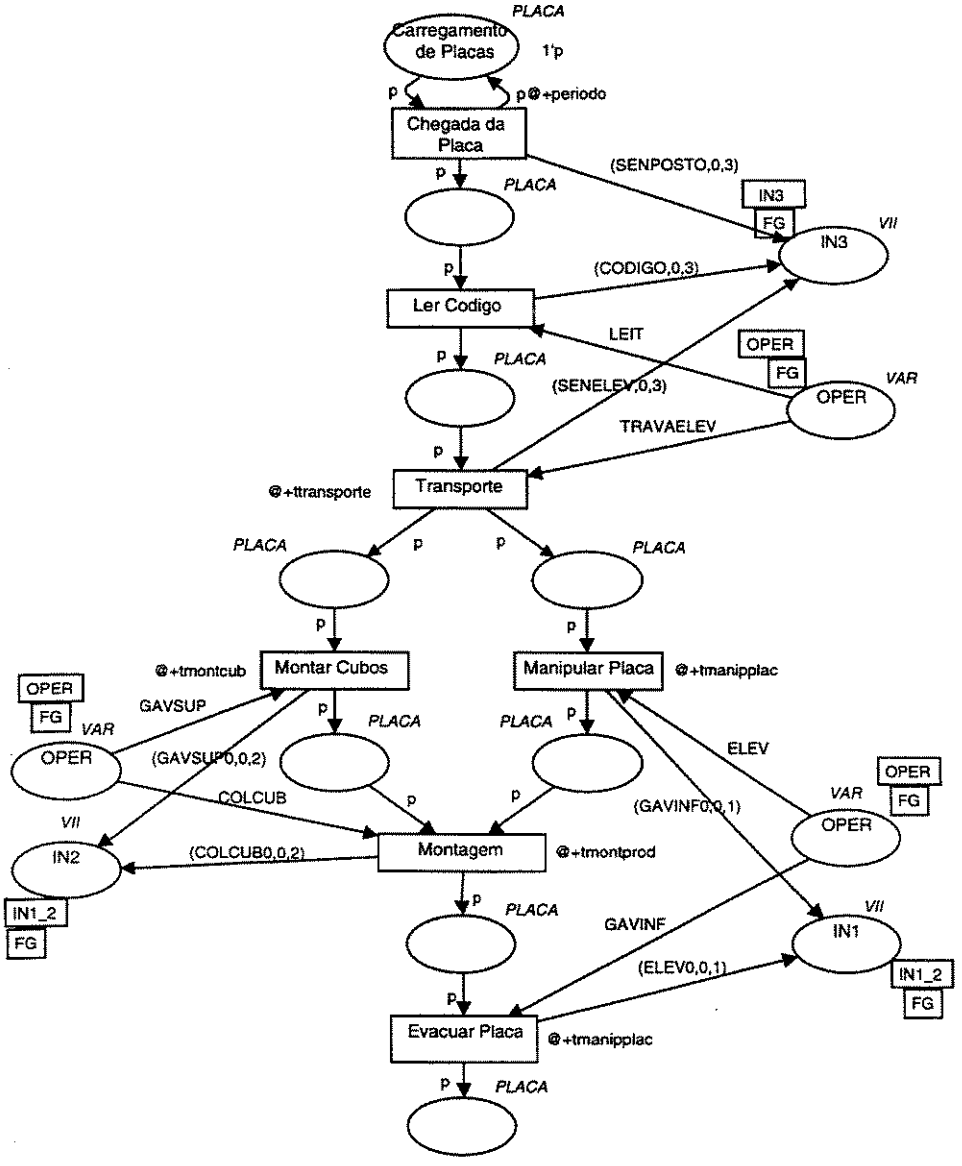
ANEXO C: MODELOS DINÂMICOS

C.1. ARQUITETURA A

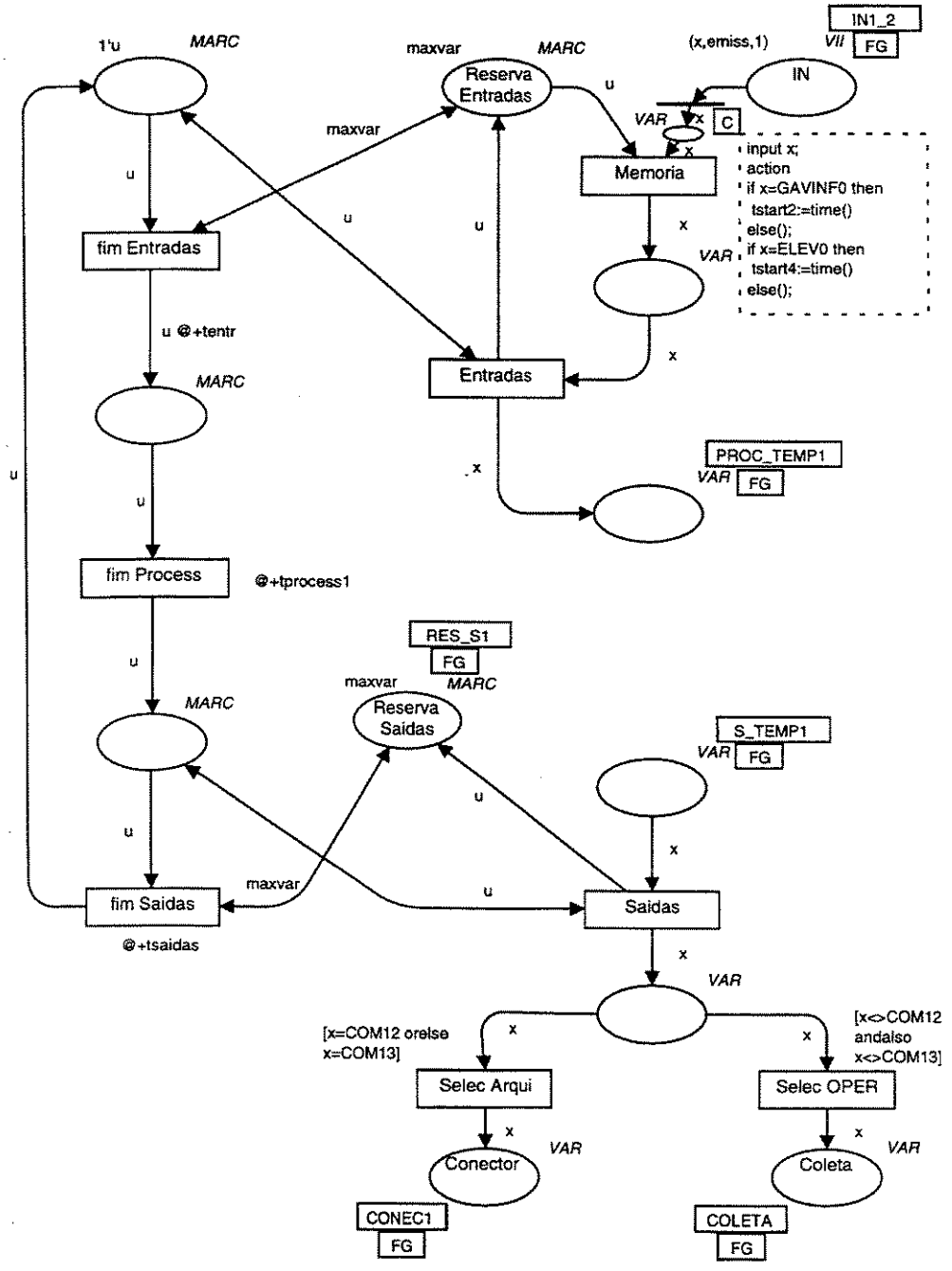
C.1.1. QUADRO DE DECLARAÇÕES A

```
val tentr = 1;
val tprocess1 = 29;
val tprocess2 = 31;
val tprocess3 = 47;
val tsaidas = 1;
val tpoll = 23;
val ttrans = 103;
val tack = 11;
val periodo = 30000;
val ttransporte = 9000;
val tmanipplac = 5000;
val tmontcub = 4000;
val tmontprod = 3000;
val idmax = 3;
val nmaxescrav = 2;
val mestre = 3;



color MARC = with u timed;
color VAR = with COM12| COM13| COM21| COM31| COM32| ELEV| ELEV0|
GAVINF| GAVINF0| GAVSUP| GAVSUP0| COLCUB| COLCUB0| SENPOSTO| LEIT|
CODIGO| TRAVAELEV | SENELEV| FIM timed;

color IDF = int with 0..idmax timed;
color VII = product VAR*IDF*IDF;
color PLACA = with p timed;
color INT = int;

val maxvar = 5`u;
val listescrav = 1`1 + 1`2;

fun prox(emiss) = if emiss=nmaxescrav then 1`1 else 1`(emiss+1);

var x, m : VAR;
var emiss, recep : IDF;

globref tstart1 = 0 : INT;
globref tstart2 = 0 : INT;
globref tstart3 = 0 : INT;
globref tstart4 = 0 : INT;
```

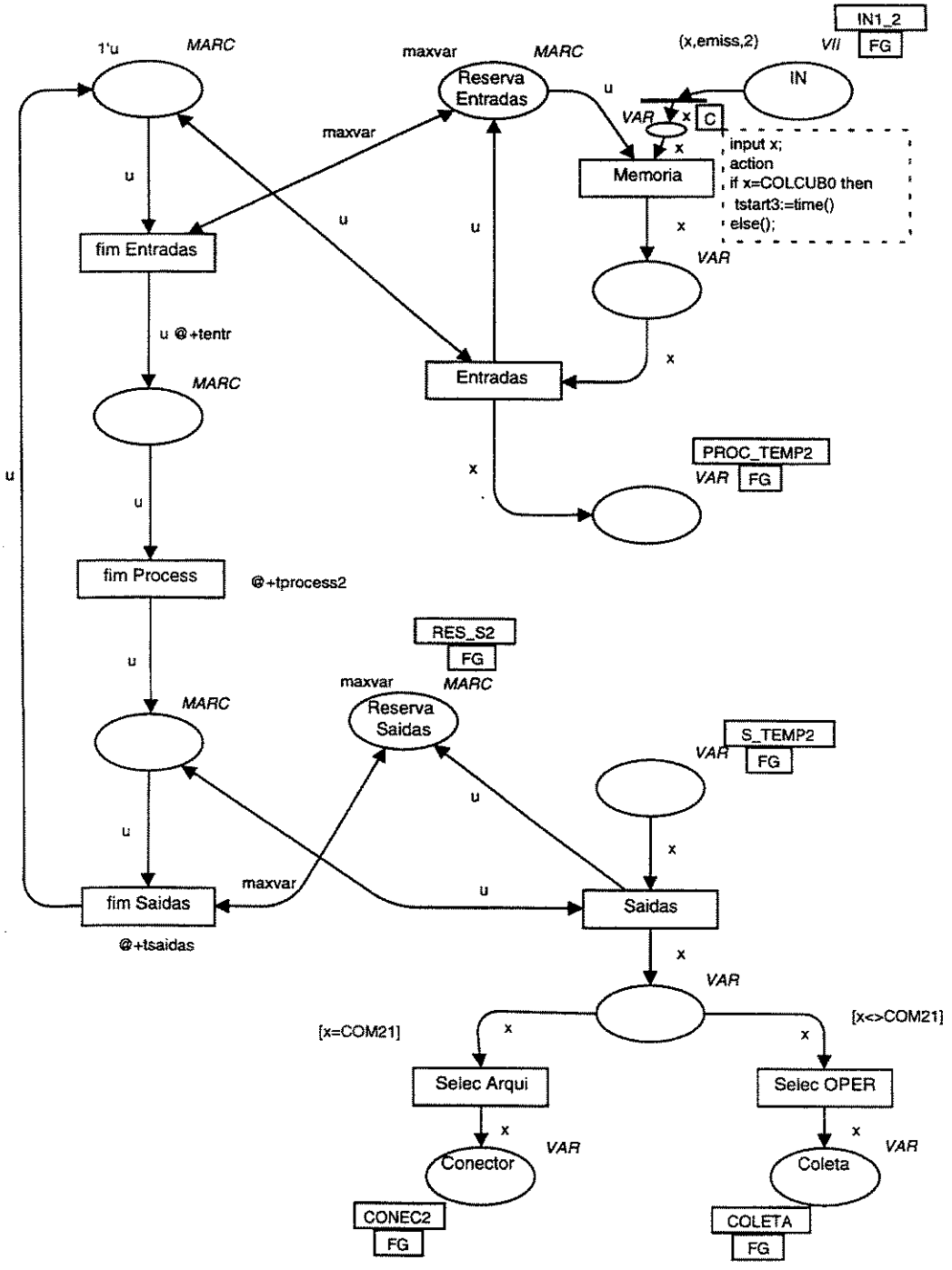
C.1.2. PARTE OPERATIVA A



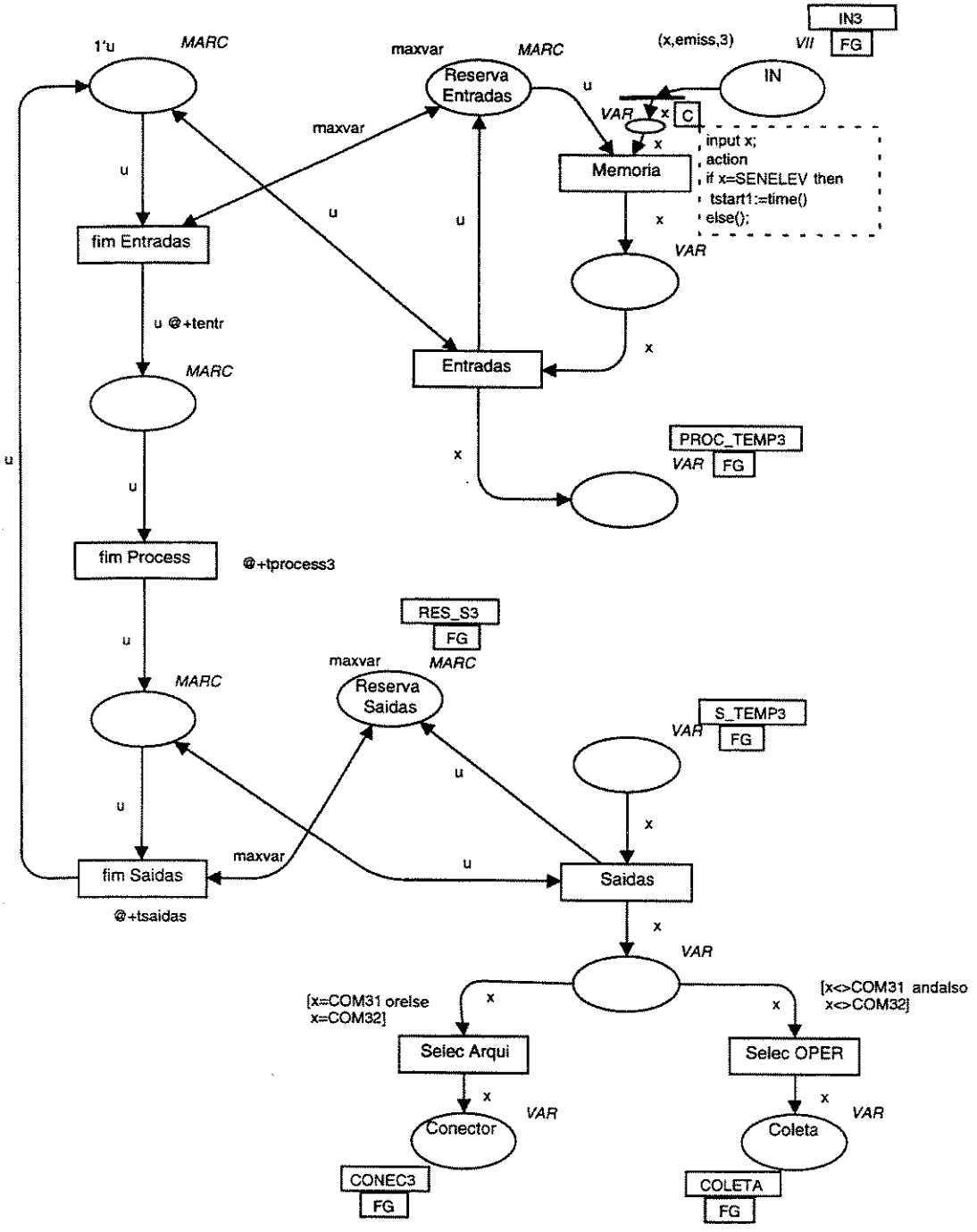
C.1.3. CLP 1 A


1'u
MARC
maxvar
MARC
Reserva Entradas
(x,emiss,1)
IN1_2
VII
FG
IN
u
VAR
x
C
maxvar
Memoria
input x;
action
if x=GAVINF0 then
tstart2:=time()
else();
if x=ELEV0 then
tstart4:=time()
else();
fim Entradas
u @+tentr
MARC
Entradas
PROC_TEMP1
FG
fim Process
@+tprocess1
RES_S1
FG
maxvar
MARC
Reserva Saidas
S_TEMP1
FG
fim Saidas
@+tsaidas
Saidas
[x=COM12 orelse x=COM13]
[x<>COM12 andalso x<>COM13]
Selec Arqui
Selec OPER
Conector
Coleta
CONEC1
FG
COLETA
FG

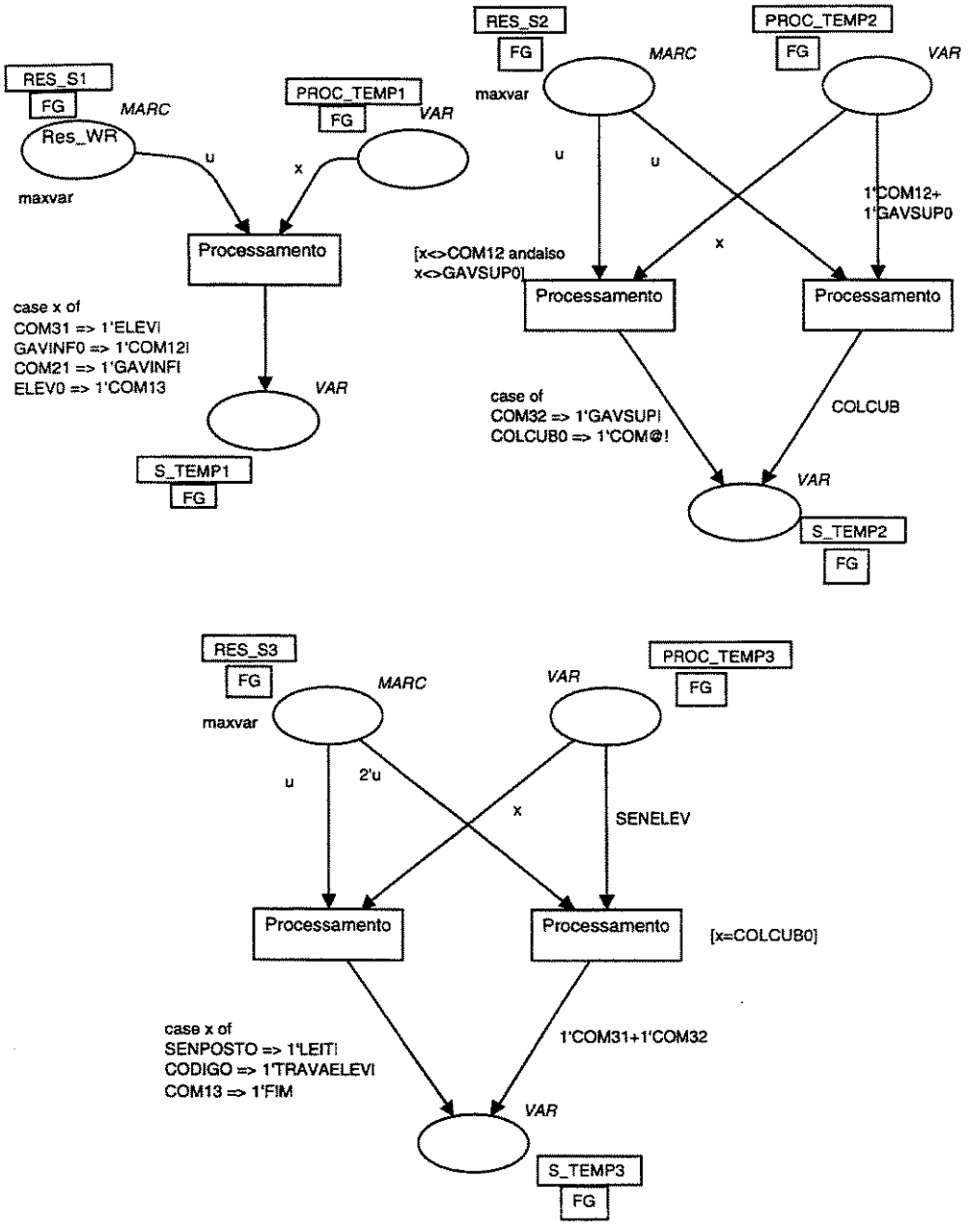
C.1.4. CLP 2 A


IN1_2
VII
FG
1'u
MARC
maxvar
MARC
(x,emiss,2)
Reserva Entradas
IN
u
VAR
x
C
maxvar
x
input x;
action
if x=COLCUB0 then
tstart3:=time()
else();
Memoria
u
u
u
x
VAR
fim Entradas
u @+tentr
MARC
Entradas
x
PROC_TEMP2
VAR
FG
u
x
u
fim Process
@+tprocess2
RES_S2
FG
maxvar
MARC
u
MARC
Reserva Saidas
S_TEMP2
VAR
FG
u
u
x
maxvar
u
fim Saidas
Saidas
@+tsaidas
x
VAR
[x=COM21]
[x<>COM21]
x
x
Selec Arqui
Selec OPER
x
x
VAR
VAR
Conector
Coleta
CONEC2
FG
COLETA
FG

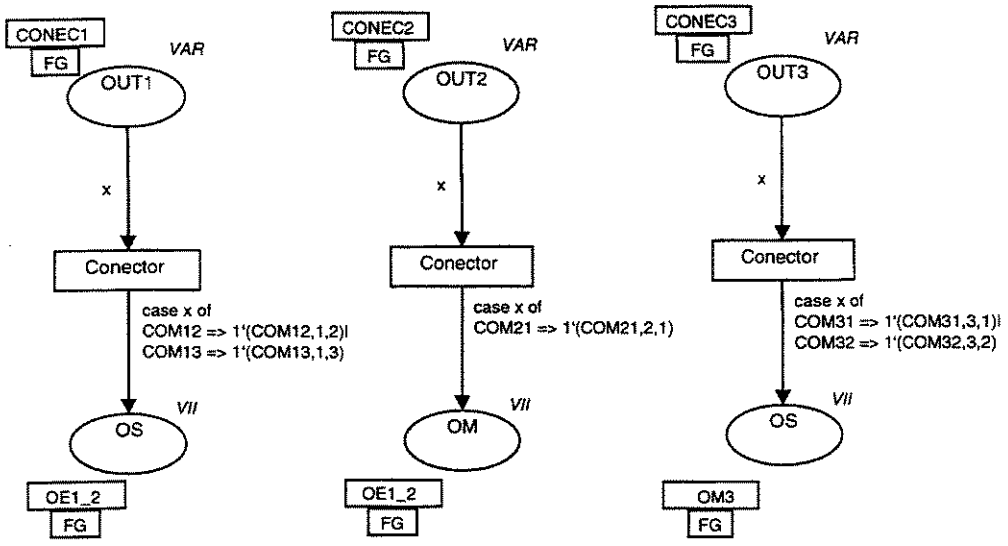
C.1.5. CLP 3 A


IN3
VII
FG
1'u
MARC
maxvar
MARC
(x,emiss,3)
IN
Reserva Entradas
u
VAR
x
C
maxvar
x
Memoria
input x;
action
if x=SENELEV then
tstart1:=time()
else();
u
u
u
x
VAR
fim Entradas
u @+tentr
MARC
Entradas
x
PROC_TEMP3
VAR
FG
u
x
u
fim Process
@+tprocess3
RES_S3
FG
u
maxvar
MARC
MARC
Reserva Saidas
S_TEMP3
VAR
FG
u
u
x
maxvar
u
fim Saidas
Saidas
@+tsaidas
x
VAR
[x=COM31 orelse x=COM32]
x
x
[x<>COM31 andalso x<>COM32]
Selec Arqui
Selec OPER
x
x
VAR
VAR
Conector
Coleta
CONEC3
FG
COLETA
FG

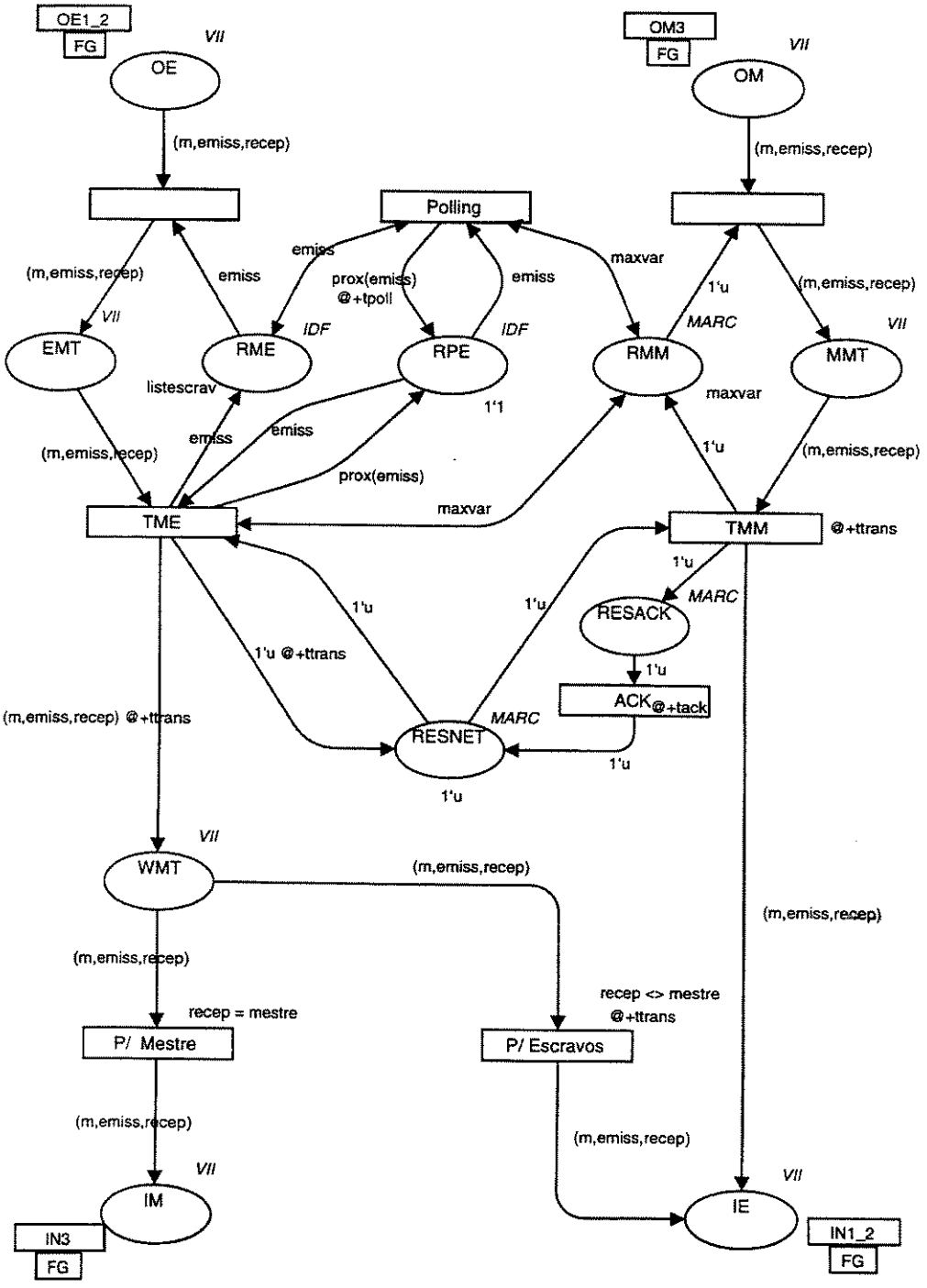
C.1.6. PROCESSAMENTO A



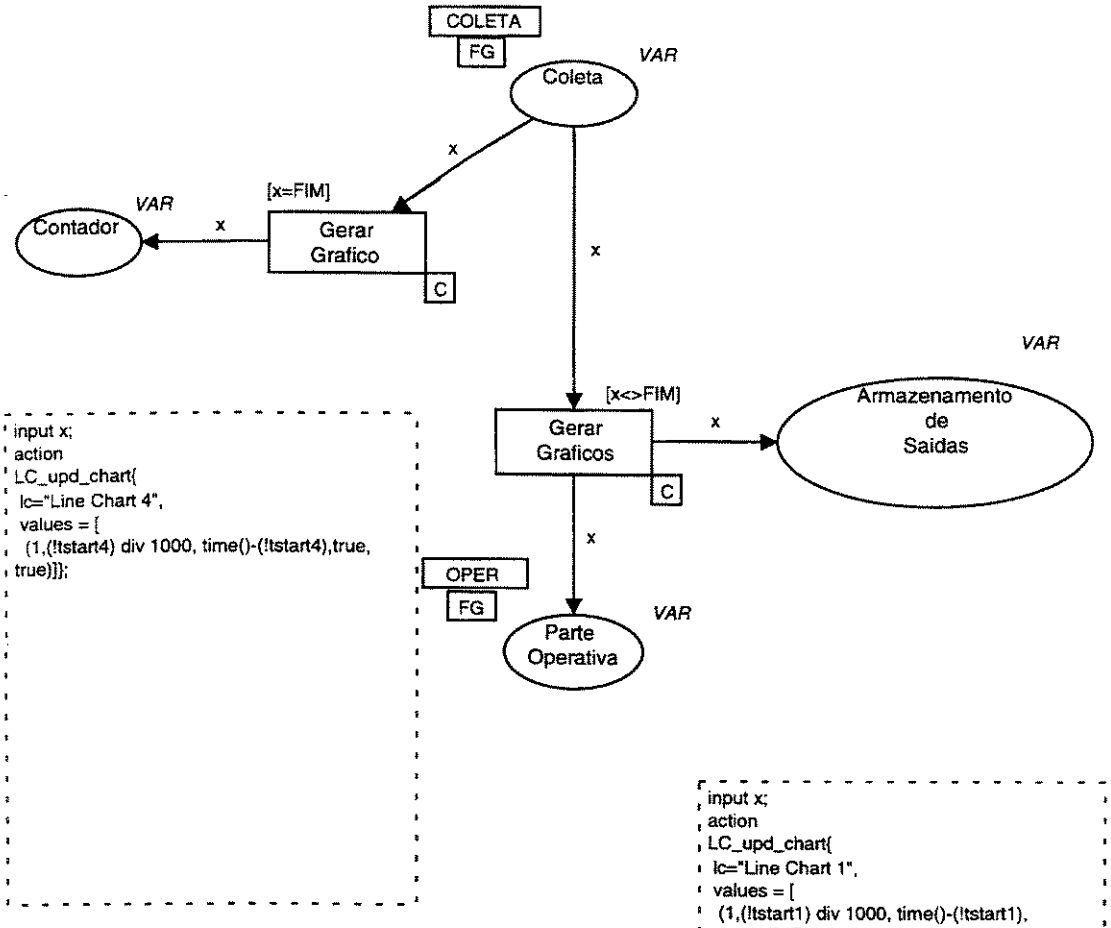
C.1.7. CONECTORES A



C.1.8. REDE UNITELWAY A



C.1.9. COLETA A




```
input x;
action
LC_upd_chart(
 lc="Line Chart 4",
 values = [
  (1,(!tstart4) div 1000, time()-(!tstart4),true,
true)]);
```

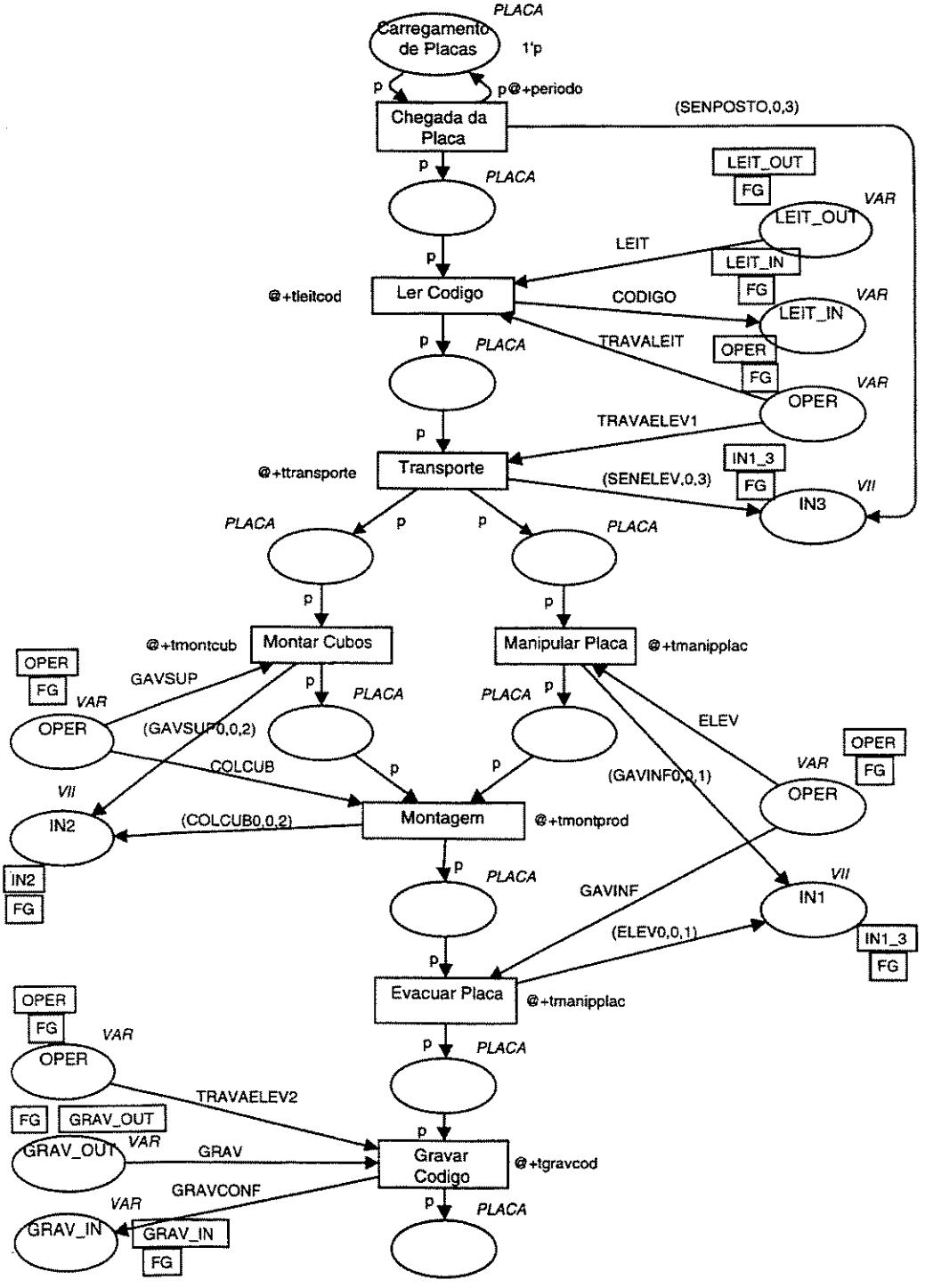
```
input x;
action
LC_upd_chart(
 lc="Line Chart 1",
 values = [
  (1,(!tstart1) div 1000, time()-(!tstart1),
true,x=ELEV),
  (2,(!tstart1) div 1000, time()-(!tstart1),
true,
x=GAVSUP)]);

LC_upd_chart(
 lc="Line Chart 2",
 values = [
  (1,(!tstart2) div 1000, time()-(!tstart2),
true,x=COLCUB)]);

LC_upd_chart(
 lc="Line Chart 3",
 values = [
  (1,(!tstart3) div 1000, time()-(!tstart3),
true,x=GAVINF)]);
```

C.2. ARQUITETURA B

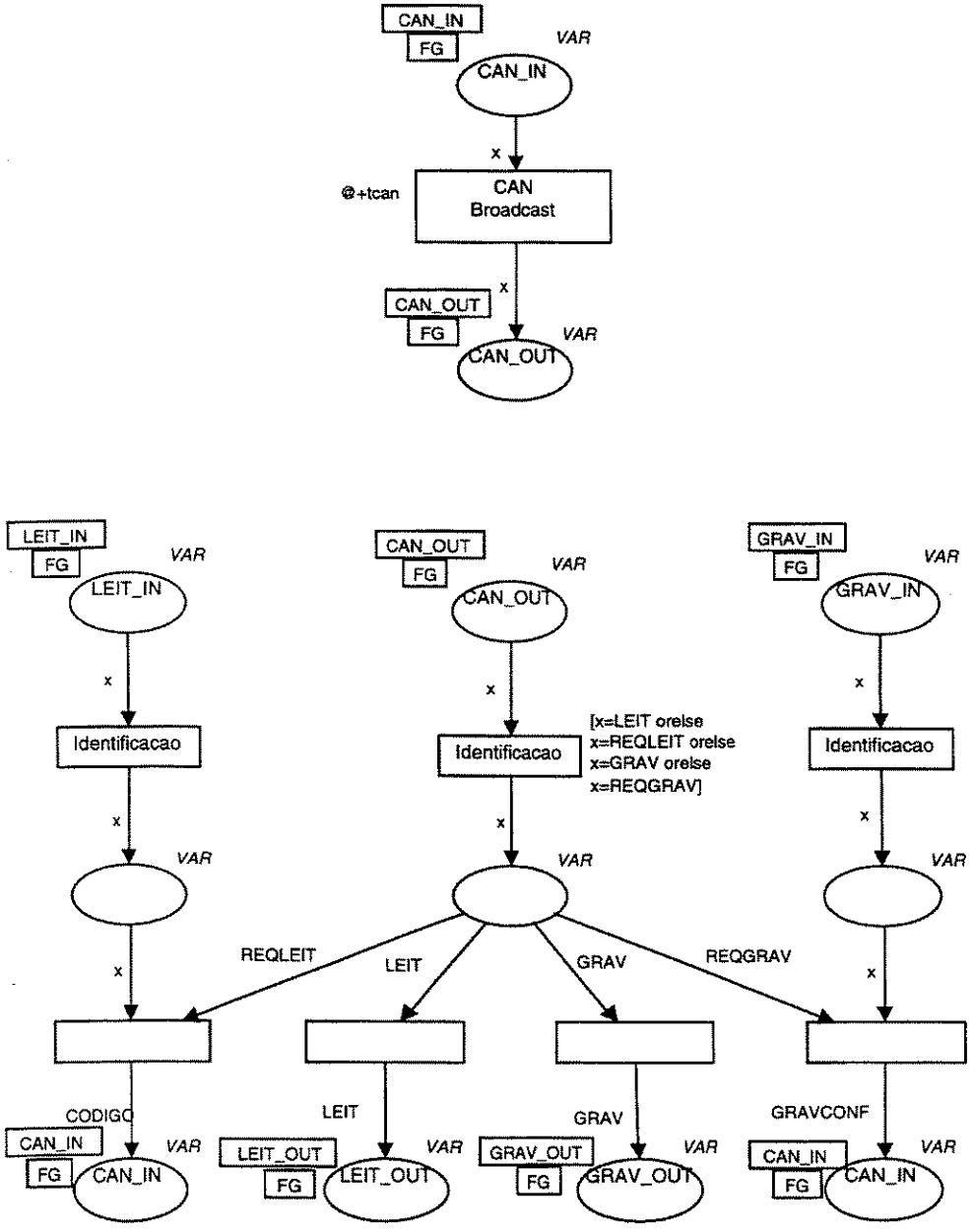
C.2.1. QUADRO DE DECLARAÇÕES B

```
val tentr = 1;
val tprocess1 = 29;
val tprocess2 = 31;
val tsaidas = 1;
val tpoll = 23;
val ttrans = 103;
val tack = 11;
val tbalogh = 20;
val tcan = 10;
val periodo = 30000;
val tleitcod = 25;
val ttransporte = 10000;
val tmanipplac = 5000;
val tmontcub = 4000;
val tmontprod = 3000;
val tgravcod = 25;
val idmax = 3;
val nmaxescrav = 2;
val mestre = 2;

color MARC = with u timed;
color VAR = with COM12| COM13| COM21| COM23| COM31| COM32| ELEV| ELEV0|
GAVINF| GAVINF0| GAVSUP| GAVSUP0| COLCUB| COLCUB0| SENPOSTO| TRAVALEIT| LEIT|
REQLEIT| CODIGO| TRAVAELEV1| SENELEV| GRAV| REQGRAV| TRAVAELEV2| GRAVCONF| FIM
timed;
color IDF = int with 0..idmax timed;
color VII = product VAR*IDF*IDF;
color PLACA = with p timed;
color INT = int;

val maxvar = 5`u;
val listescrav = 1`1 + 1`3;

fun prox(emiss) = if emiss=nmaxescrav+1 then 1`1 else 1`(emiss+2);

var x, m : VAR;
var emiss, recep : IDF;

globref tstart1 = 0 : INT;
globref tstart2 = 0 : INT;
globref tstart3 = 0 : INT;
globref tstart4 = 0 : INT;
```

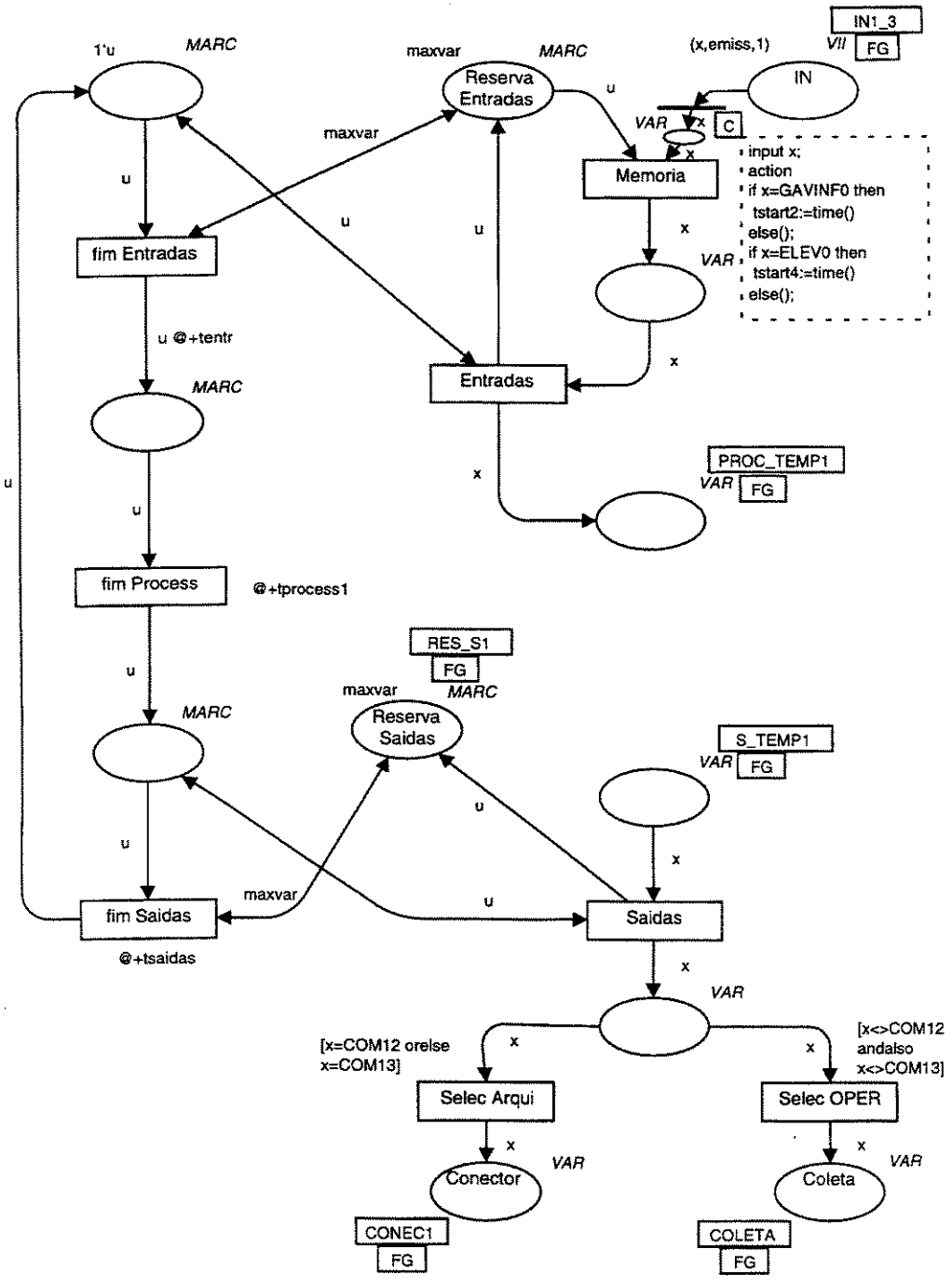
C.2.2. PARTE OPERATIVA B



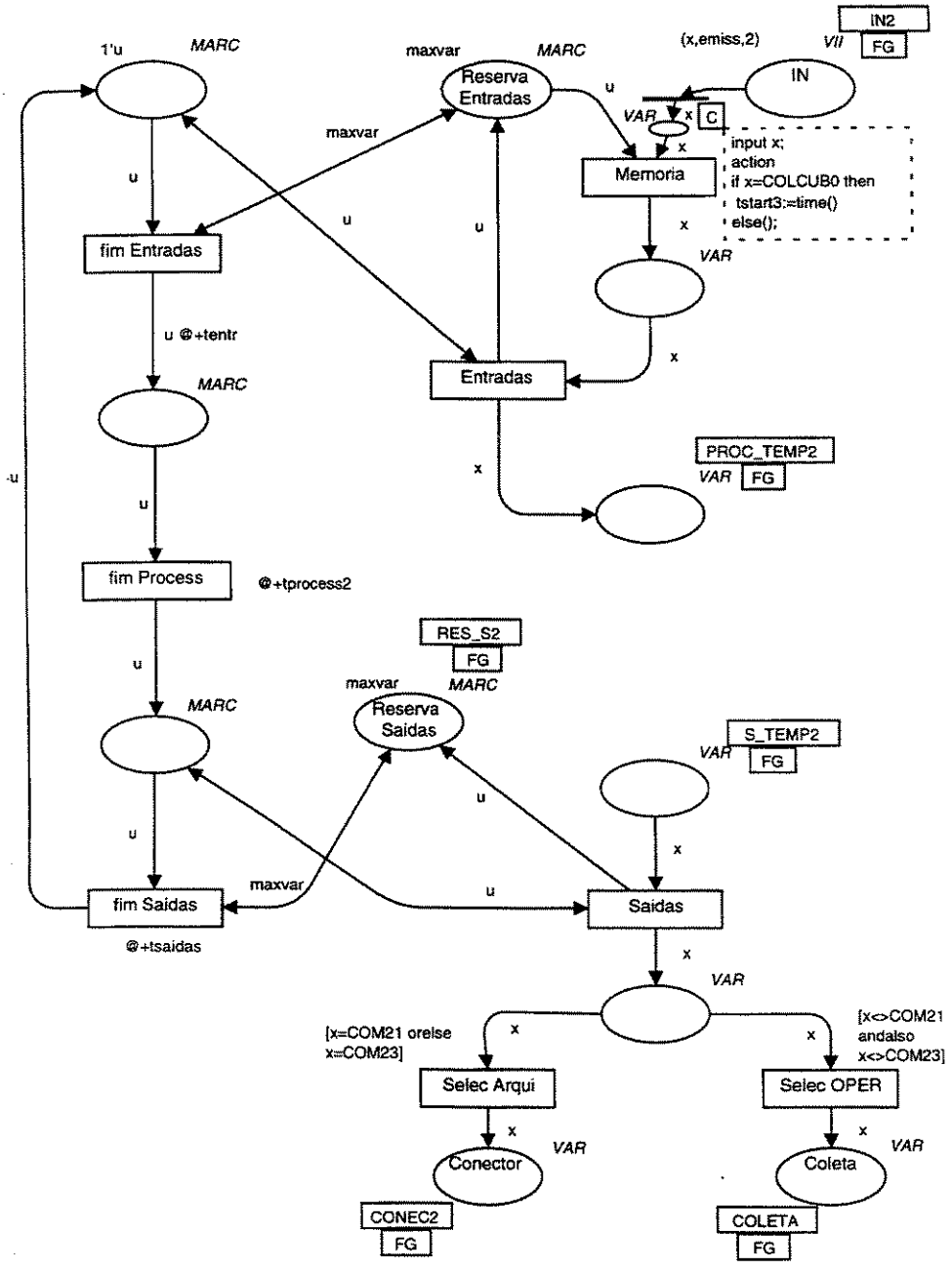
C.2.3. BARRAMENTO CAN B



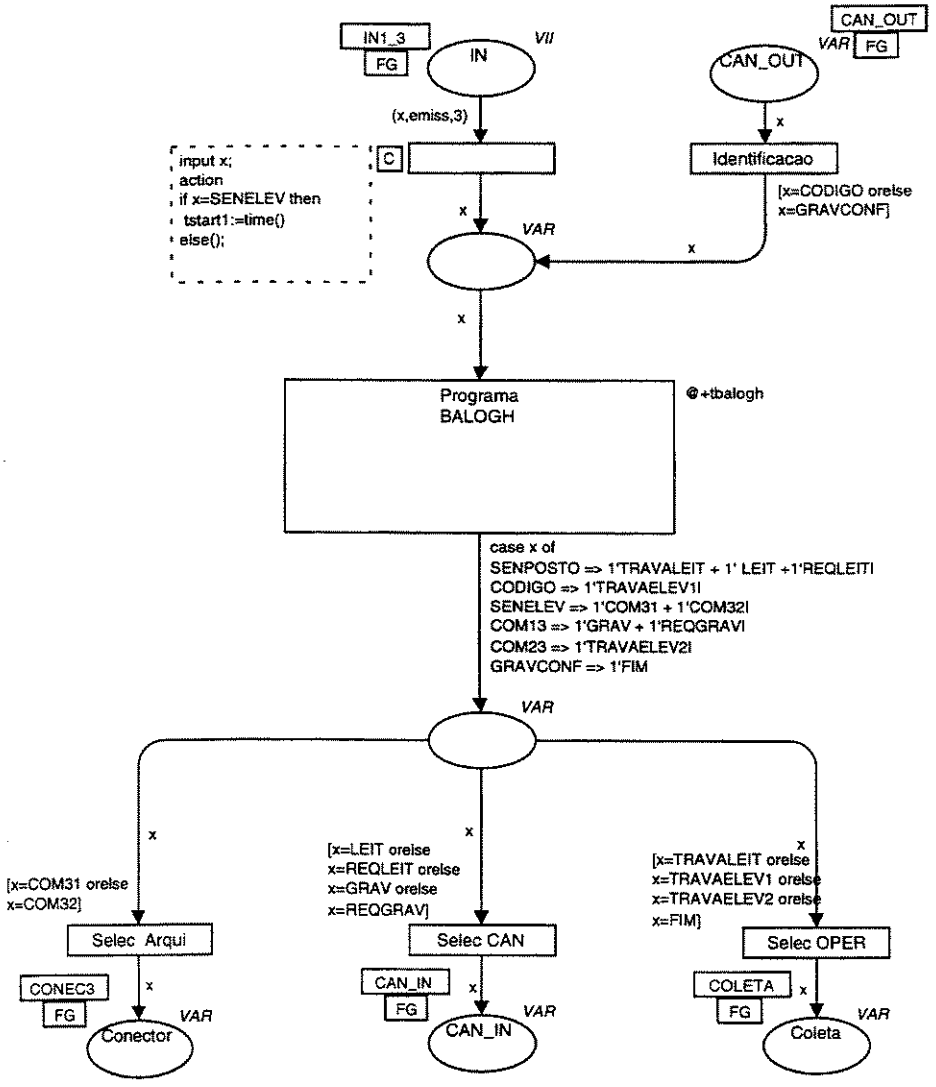
C.2.4. CLP 1 B


1'u
MARC
maxvar
MARC
Reserva Entradas
(x,emiss,1)
IN1_3
VII
FG
IN
u
VAR
x
C
maxvar
Memoria
input x;
action
if x=GAVINF0 then
tstart2:=time()
else();
if x=ELEV0 then
tstart4:=time()
else();
fim Entradas
u @+tentr
MARC
Entradas
VAR
PROC_TEMP1
FG
fim Process
@+tprocess1
RES_S1
FG
maxvar
MARC
Reserva Saidas
S_TEMP1
fim Saidas
@+tsaidas
Saidas
[x=COM12 orelse x=COM13]
[x<>COM12 andalso x<>COM13]
Selec Arqui
Selec OPER
Conector
Coleta
CONEC1
COLETA

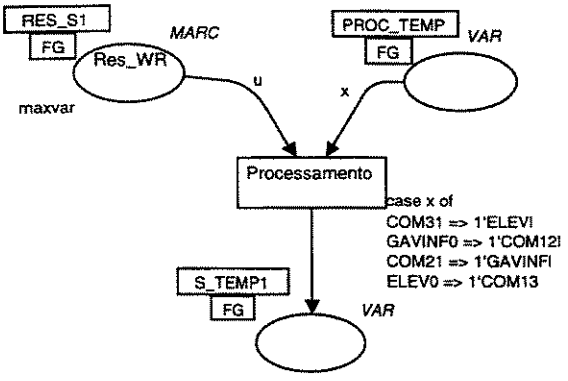
C.2.5. CLP 2 B

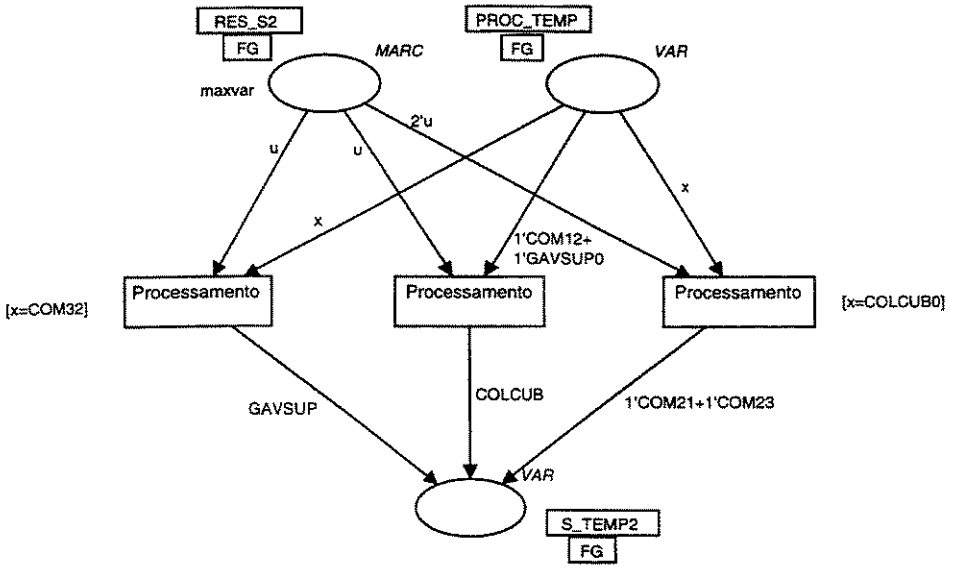

1'u
MARC
maxvar
MARC
Reserva Entradas
(x,emiss,2)
IN2
VII
FG
IN
u
VAR
x
C
input x;
action
if x=COLCUB0 then
tstart3:=time()
else();
Memoria
maxvar
u
u
u
x
VAR
fim Entradas
u @+tentr
MARC
Entradas
x
PROC_TEMP2
VAR
FG
u
fim Process
@+tprocess2
RES_S2
FG
maxvar
MARC
Reserva Saidas
S_TEMP2
VAR
FG
u
u
x
maxvar
u
fim Saidas
Saidas
@+tsaidas
x
VAR
[x=COM21 orelse x=COM23]
[x<>COM21 andalso x<>COM23]
Selec Arqui
Selec OPER
x
VAR
Conector
Coleta
CONEC2
FG
COLETA
FG

C.2.6. PLACA PROGRAMÁVEL B

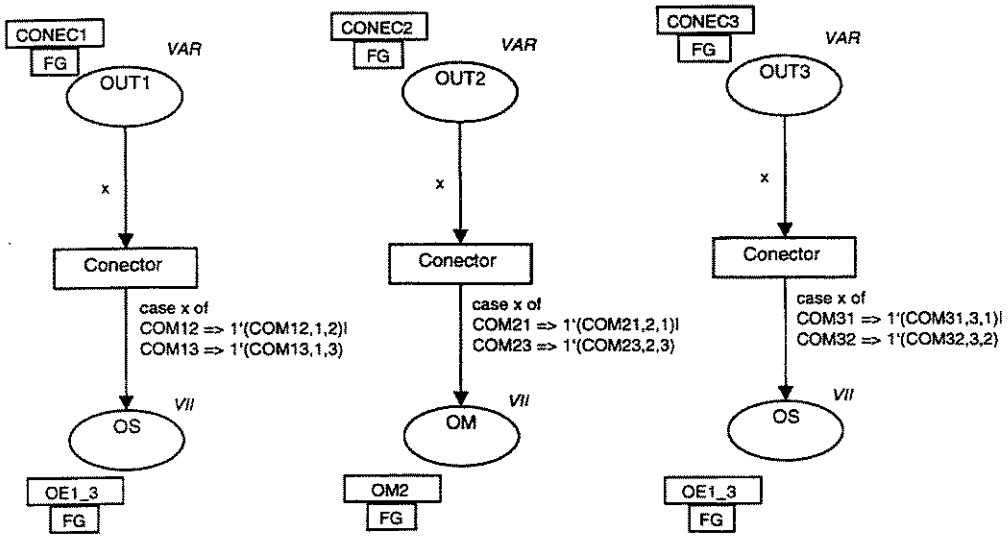


C.2.7. PROCESSAMENTO B

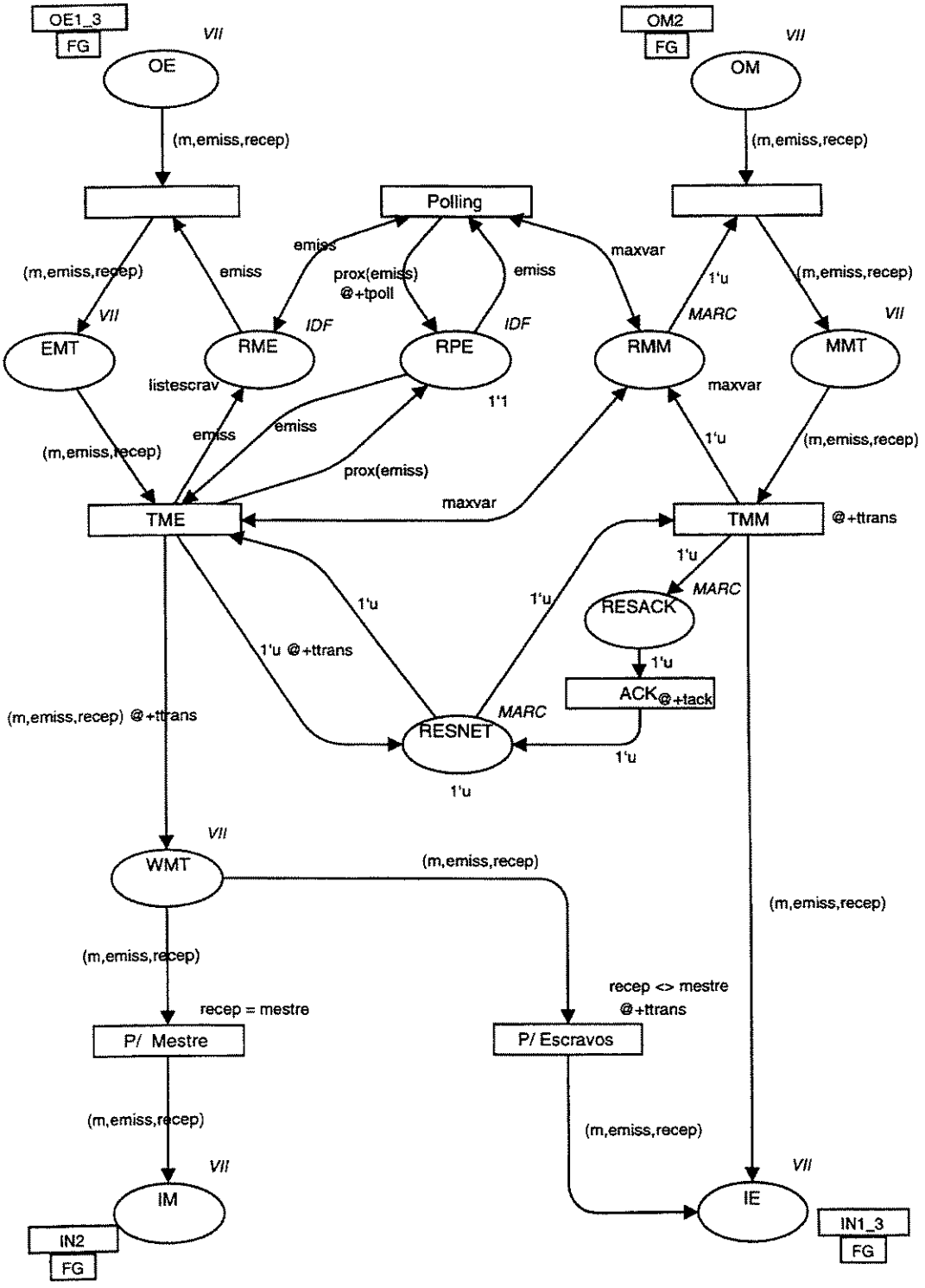




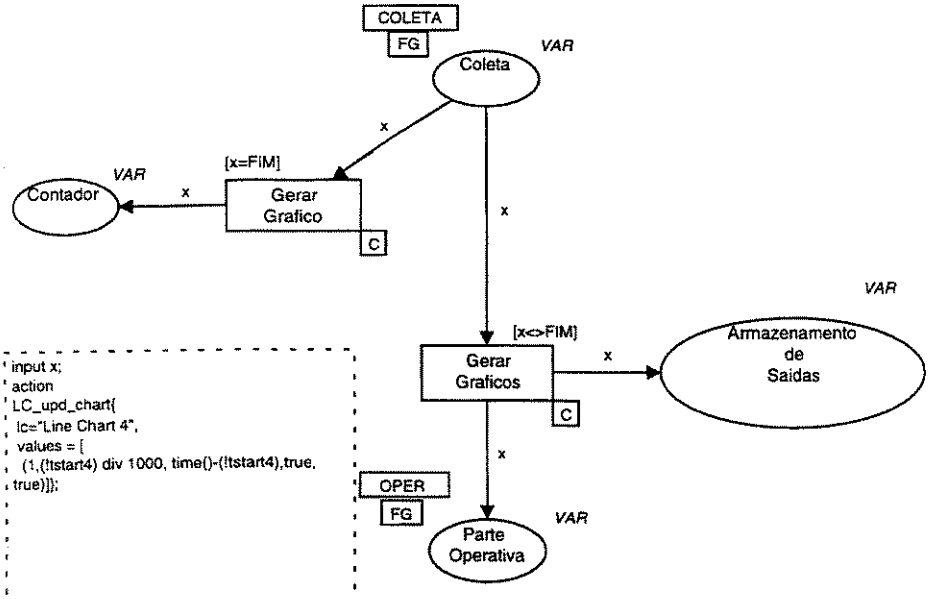
C.2.8. CONECTORES B



C.2.9. REDE UNITELWAY B



C.2.10. COLETA B




```
input x;
action
LC_upd_chart{
 lc="Line Chart 4",
 values = [
  (1,(!tstart4) div 1000, time()-(!tstart4),true,
true)]};
```

```
input x;
action
LC_upd_chart{
 lc="Line Chart 1",
 values = [
  (1,(!tstart1) div 1000, time()-(!tstart1),
true,x=ELEV),
  (2,(!tstart1) div 1000, time()-(!tstart1),
true,
x=GAVSUP)]};

LC_upd_chart{
 lc="Line Chart 2",
 values = [
  (1,(!tstart2) div 1000, time()-(!tstart2),
true,x=COLCUB)]};

LC_upd_chart{
 lc="Line Chart 3",
 values = [
  (1,(!tstart3) div 1000, time()-(!tstart3),
true,x=GAVINF),
  (2,(!tstart3) div 1000, time()-(!tstart3),
true,
x=TRAVAELEV2)]};
```