PEDRO MANUEL GONZÁLEZ DEL FOYO

GHENESYS : UMA REDE ESTENDIDA ORIENTADA A OBJETOS PARA PROJETO DE SISTEMAS DISCRETOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia.

São Paulo
2001

PEDRO MANUEL GONZÁLEZ DEL FOYO

GHENESYS: UMA REDE ESTENDIDA ORIENTADA A OBJETOS PARA O PROJETO DE SISTEMAS DISCRETOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia.

Área de Concentração:
Engenharia Mecânica - Mecatrônica

Orientador:
José Reinaldo Silva

São Paulo
2001

AGRADECIMENTOS

Ao amigo e orientador Prof. Dr. José Reinaldo Silva pela constante orientação e apoio durante o desenvolvimento deste trabalho.

A minha esposa Mariblanca Alonso pelo seu estímulo e compreensão.

Aos meus colegas de trabalho do departamento de Informática da Universidade de Oriente, principalmente ao chefe do departamento MSc. José Cuza Freyre pelo seu apoio para a culminação dos meus estudos de pós-graduação.

A todos aqueles que direta ou indiretamente, colaboraram na execução deste trabalho.

Candidato: Pedro Manuel González del Foyo

Título do Trabalho: "GHENeSys: Uma rede estendida orientada a objetos para projeto de sistemas discretos".

Orientador: Prof. Dr. José Reinaldo Silva.

Data da Defesa: 27 de agosto 2001

ERRATA

Na Lista de Símbolos

Acrescentar os seguintes símbolos:

$[\mathbf{M}>$ Conjunto das marcações alcançáveis a partir de **M**.

$\exists|$ Existe ao menos um.

No Resumo

Na linha 13, onde se lê "Isto se utilizarmos uma versão orientada a objetos para redes de Petri", leia-se "Isto ocorre quando se utiliza uma versão orientada a objetos para redes de Petri".

Na linha 17, onde se lê "foi chamado de GHENeSys", leia-se "é chamado de GHENeSys".

Na linha 28, onde se lê "embora tenhamos ainda um modelo preliminar", leia-se "embora tenha-se ainda apenas um modelo preliminar".

Pág. 2

Na linha 3, onde se lê "ficou", leia-se "fica".

Pág. 4

Na linha 19, onde se lê "baseado", leia-se "baseadas".

Pág. 5

Na linha 5, onde se lê "Esta rede resultará a base para a implementação de um sistema", leia-se "Esta rede resultará na base para a implementação de um sistema".

Pag. 6

Na linha 5, onde se lê "do tipo de aplicações" leia-se "sobre os tipos de aplicações".

Pag. 7

Retirar o parágrafo primeiro.

Na linha 6, onde se lê "Algumas abordagens são freqüentemente utilizadas nestes trabalhos" leia-se "Algumas abordagens da área de Ciências da Computação são freqüentemente utilizadas nestes trabalhos"

Pag. 8

Na linha 8, onde se lê "constituem uma limitação das redes de Petri [Esser, 97]." leia-se "constituem uma limitação das redes de Petri [Esser, 97], apesar de já existirem as redes de Petri de alto nível."

Na linha 16, onde se lê "Justamente a partir do ano 95 começaram a aparecer propostas de unificação destas duas tendências" leia-se "A partir do ano 95 começa a parecer uma nova tendência que envolve as propostas de unificação destas duas tendências".

Pag. 10

Na linha 3, onde se lê "A unificação da abordagem ODRP e RPDO" leia-se "A unificação das abordagens ODRP e RPDO".

Pag. 12

Na linha 17, onde se lê "pêlos" leia-se "pelos".

Pag. 13

Na linha 12, onde se lê "esta verificação só é possível uma vez concluído o modelo" leia-se "esta verificação só é possível uma vez construído todo o modelo".

Pag. 16

Na linha 10, onde se lê "regra de disparo da rede" leia-se "regra de disparo das transições da rede".

Pag. 17

Na linha 15, onde se lê "Considerasse que isto é um primeiro passo" leia-se "Considera-se isto como um primeiro passo".

Pag. 20

Na linha 15, onde se lê "se fala de "*design top-down*"" leia-se "se considera a abordagem de "*design top-down*"".

Na linha 19, onde se lê "para apoiar a análise" leia-se "visando apoio a análise".

Pag. 29

Na linha 11, onde se lê "A seguir faremos uma breve introdução destes modelos de redes usando a terminologia utilizada em [Reisig, 82]" leia-se "A introdução destes modelos de redes é feita usando-se a terminologia adotada em [Reisig, 82]"

Pag. 30

Na linha 9, acrescentar $\forall x \in \mathbf{X}$.

Pag. 31

Na linha 2, onde se lê "O subconjunto $c \in B$ é chamado *case*" leia-se " O subconjunto $c \subseteq B$ é chamada de *case*"

Pag. 33

Na linha 5, acrescentar "$\forall (s,t) \in \mathbf{F} \wedge \forall (t,s) \in \mathbf{F}$"

Pag. 34

Na linha 9, onde se lê "dado um número k $\in \mathbb{N}^+$ " leia-se "dado um número $k \in \mathbb{N}^+$, $k > 1$".

Pag. 39

Na linha 15, onde se lê "Portanto seria interessante ter uma abordagem *top down*" leia-se "Portanto é interessante considerar uma abordagem *top down*"

Pag 43.

Substituir a Definição 1 pela seguinte definição:

DEFINIÇÃO 1. Uma rede GHENeSys é uma sextupla **N=(L, A, F, K, M,** Π**)** onde:

i) $\mathbf{L} \cap \mathbf{A} = \varnothing$

ii) $\mathbf{L} \cup \mathbf{A} \neq \varnothing$

iii) $\mathbf{F} \subseteq (\mathbf{L}\mathbf{x}\mathbf{A}) \cup (\mathbf{A}\mathbf{x}\mathbf{L})$

iv) $\mathbf{L} = \mathbf{B} \cup \mathbf{P}$

v) $\mathbf{P} = \mathbf{H} \cup \mathbf{I}$

vi) $\mathbf{K}: \mathbf{L} \rightarrow \mathbb{N}^+$ e para todo $p \in \mathbf{P}$, $K(p) = 1$.

vii) $\mathbf{M}: \mathbf{L} \rightarrow \mathbb{N}^+$ sendo que: $\mathbf{M}(l) \leq \mathbf{K}(l)$ para todo $l \in \mathbf{L}$

viii) $\Pi: (\mathbf{B} \cup \mathbf{A}) \rightarrow \{0,1\}$

Pag. 44

Acrescentar ao final do primeiro parágrafo "O conjunto **P** (*pseudoboxes*) é formado por arcos habilitadores (**H**) e arcos inibidores (**I**). Π é uma função que identifica os elementos macros dentro da rede, sendo igual a zero para elementos simples e igual a um para os elementos macros."

Pag. 45

Eliminar definição de **F**

Pag. 46

Na linha 12, onde se lê "$\forall b \in {}^{\bullet}a,\ b \in \mathbf{B} \wedge b \in \mathbf{M}$" leia-se "$\forall b \in {}^{\bullet}a,\ b \in \mathbf{B} \mid M(b) \geq 1$"

Pag. 47

Na linha 3, substituir o item iii) pelo seguinte:

iii) Uma marcação **M** que habilita a leva a uma marcação **M'**, onde:

$$\mathbf{M'} = \begin{cases} \mathbf{M}(l)\text{-}1 \text{ para } \forall l \in \mathbf{B} \mid l \in {}^{\bullet}a \text{ e } \forall g \in a^{\bullet},\ \mathbf{M}(g) < \mathrm{K}(g). \\ \mathbf{M}(l)\text{+}1 \text{ para } \forall l \in \mathbf{B} \mid l \in a^{\bullet} \text{ e } \mathbf{M}(l) < \mathrm{K}(l). \\ \mathbf{M}(l) \quad \text{para o resto dos casos.} \end{cases}$$

e pode ser representado como $\mathbf{M}[a>\mathbf{M'}$. Isto indica que a marcação persistente dos *pseudoboxes* não é alterada com o disparo.

Pag. 49

Substituir a definição da matriz de incidência pela seguinte:

Sendo assim, a matriz de incidência $C:(LxA) \rightarrow \{-1,0,1\}$ de **N** seria definida por:

$$C(l,a) = \begin{cases} 1 \text{ se } (b,a) \notin \mathbf{F}, (a,b) \in \mathbf{F} \\ -1 \text{ se } (b,a) \in \mathbf{F} \text{ e } (a,b) \notin \mathbf{F} \\ 1 \text{ se } (p,a) \in \mathbf{F} \text{ e p é um arco inibidor} \\ -1 \text{ se } (p,a) \in \mathbf{F} \text{ e p é um arco habilitador} \\ 0 \text{ para o resto dos casos} \end{cases}$$

Pag. 53

Na linha 2, onde se lê "que estamos analisando gostaríamos de saber" leia-se "que se está analisando é necessário saber"

Pag. 57

Na linha 7, onde se lê "dado um número $k \in \mathbb{N}^{+}$" leia-se "dado um número $k \in \mathbb{N}^{+}$, $k > 1$"

Pag. 60

Na linha 5 onde se lê "um elemento de singular importância na hora de modelar transferência de informação" leia-se "um elemento de singular importância na modelagem da transferência de informação"

Pag. 85

Na linha 8, onde se lê "O primeiro grupo está formado" leia-se "O primeiro grupo é formado"

Pag. 86

Na linha 6, onde se lê "O segundo grupo está formado" leia-se "O segundo grupo é formado"

Pag. 89

Na linha 8, onde se lê "comportamentos inadequados teremos devidamente identificado onde procurar e resolver estes problemas." leia-se "comportamentos inadequados isto é facilmente idenficado."

Pag 90

Eliminar os dois últimos parágrafos desta página.

Pag. 91

Substituir a figura 5.1 pela seguinte:

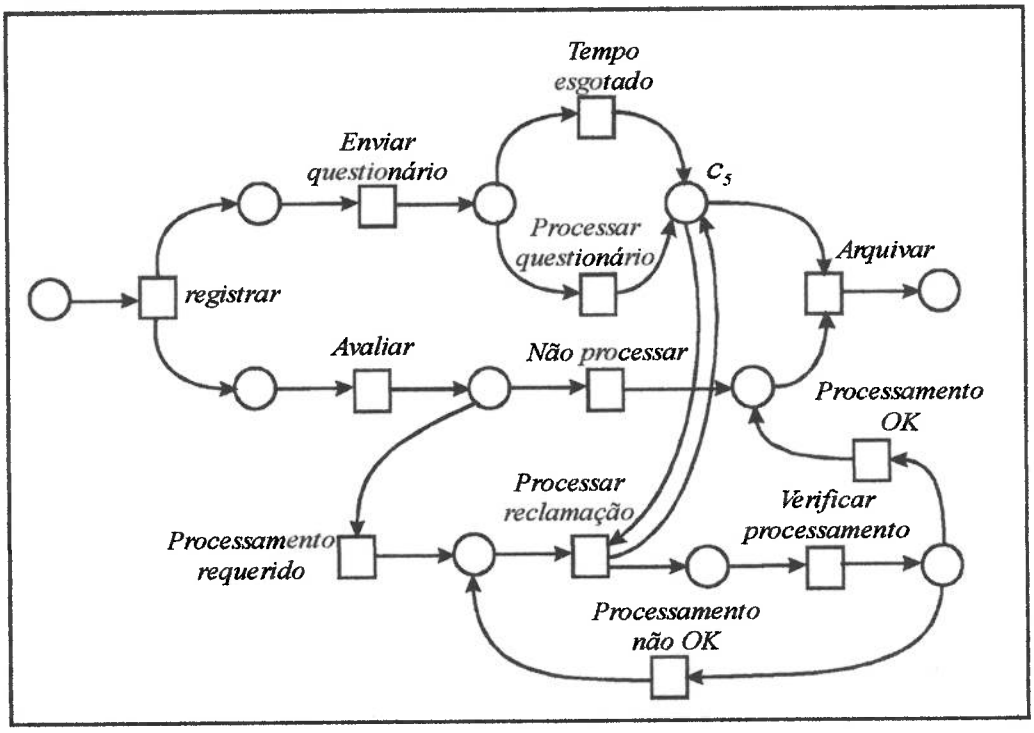


Fig 5.1 Modelagem do processamento de reclamações usando redes de Petri.

Pag. 92

Substituir a figura 5.2 pela seguinte figura:

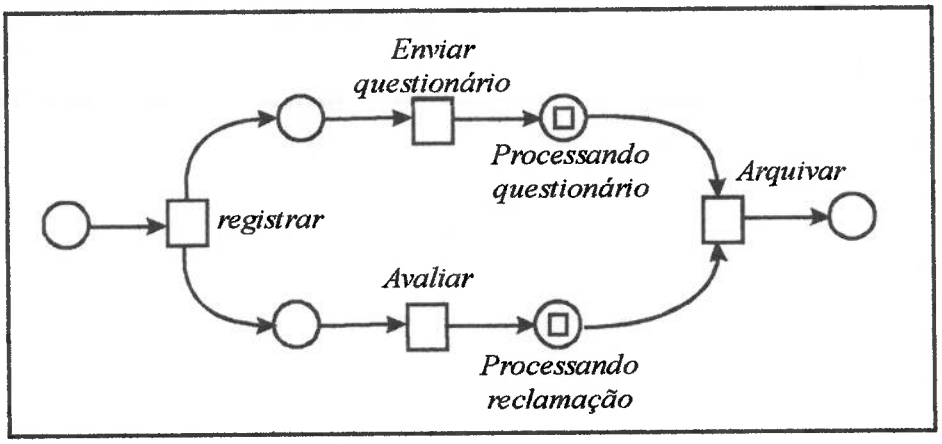


Fig. 5.2 Modelagem do processamento de reclamações no seu nível mais abstrato com a rede GHENeSys.

Pag 105

Na linha 1 onde se lê "o sistema supervisor rodará em um computador pessoal" leia-se "o sistema supervisor é implementado num computador pessoal"

Pag 116

Na linha 3, onde se lê "necessaria" leia-se "necessária"

Pag. 117

Na linha 17, onde se lê "distriuido" leia-se "distribuido"

Na linha 19, onde se lê "a análise estes diferentes sistemas de controle" leia-se "a análise destes diferentes sistemas de controle"

Pag 118

Na linha 3, onde se lê "consideravél" leia-se "considerável"

Na linha 18, onde se lê "académico" leia-se "acadêmico"

Pag 119

Na linha 6, onde se lê "contricuição" leia-se "contribuição"

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

LISTA DE SÍMBOLOS

RESUMO

ABSTRACT

1 INTRODUÇÃO.....1

1.1 MOTIVAÇÃO. 3

1.2 ORGANIZAÇÃO DO TRABALHO. 5

2 REVISÃO BIBLIOGRÁFICA. 7

2.1 ABORDAGENS DE ORIENTAÇÃO A OBJETOS EM REDES DE PETRI. 8

2.1.1 Objetos dentro de uma rede de Petri. 9

2.1.2 Redes de Petri dentro de objetos. 9

2.1.3 Unificação das tendências. 10

2.1.4 Resumo das abordagens anteriores. 14

2.2 ABORDAGENS PARA A ESTRUTURAÇÃO DAS REDES DE PETRI. 15

2.2.1 O Conceito de estrutura nas redes de Petri. *16*

2.2.2 As redes de Petri estruturadas. *17*

2.3 O USO DAS REDES DE PETRI NA MODELAGEM, ANÁLISE E PROJETO DE SISTEMAS. 18

3 TEORIAS E MODELOS BÁSICOS. 21

3.1 ESTRUTURAÇÃO EM PROGRAMAÇÃO. 21

3.2 ORIENTAÇÃO A OBJETOS 23

3.3 DEFINIÇÕES BÁSICAS SOBRE REDES DE PETRI E SUAS PROPRIEDADES. 28

3.3.1 Definições dos modelos de redes. *29*

3.3.2 Redes Elementares. *30*

3.3.3 Algumas propriedades das Redes. *33*

3.3.4 Redes estendidas *38*

3.3.5 Redes de alto nível. *40*

4 DEFINIÇÃO DA REDE GHENESYS E OS SEUS ELEMENTOS 42

4.1 DEFINIÇÃO DA REDE GHENESYS. 43

4.1.1 Matriz de incidência. *48*

4.1.2 Equação de estado. *49*

4.1.3 Algumas definições básicas para modelar comportamento na rede GHENeSys *55*

4.2 ELEMENTOS DE EXTENSÕES DA REDE GHENESYS 60

4.2.1 Hierarquia *60*

4.2.2 Orientação a objetos. *65*

5 A ESTRUTURAÇÃO NA REDE GHENESYS. **70**

5.1 ANÁLISE DAS PROPRIEDADES NA REDE GHENESYS. 71

5.1.1 Invariantes de lugar. *71*

5.1.2 Invariantes de atividade. *77*

5.1.3 Vivacidade. *82*

5.1.4 Equidade. *83*

5.2 ESTRUTURAS RELACIONADAS AO CONTROLE DE PROCESSOS. 85

5.3 APLICAÇÃO DAS REDES GHENESYS A MODELAGEM E ANÁLISE DE WORKFLOWS. 89

5.4 APLICAÇÃO DAS REDES GHENESYS À MODELAGEM E ANÁLISE DE SISTEMAS INTEGRADOS E FLEXÍVEIS. 97

6 ESTUDO DE CASO. **100**

6.1 O PROBLEMA E SUAS ESPECIFICAÇÕES. 100

6.1.1 Sistema de Segurança. *101*

6.1.2 Sistema de iluminação. *102*

6.1.3 Sistema de conforto térmico. *103*

6.1.4 A arquitetura do sistema de controle. *104*

6.2 O PROJETO DO SISTEMA. 105

6.2.1 O Modelo de dados do sistema. *106*

6.2.2 O controle das salas. *107*

6.2.3 O controle do nível superior. *110*

6.3 RESULTADOS. 114

7 CONCLUSÕES E TRABALHO FUTURO 116

7.1 CONCLUSÕES. 116

7.2 TRABALHOS FUTUROS. 118

8 REFERÊNCIAS BIBLIOGRÁFICAS. 120

APÊNDICE I – Programas em MATLAB utilizados para calcular marcações e determinar vetor de habilitação.

LISTA DE FIGURAS

Fig. 3.1 Classe Pessoa e duas de suas instâncias: Maria e Pedro.....24

Fig. 3.2 Herança: Classe Pessoa e classes derivadas.....27

Fig. 3.3. Agregação: classe Casa e as classes que a compõem.....27

Fig 4.1 Representação gráfica dos elementos que compõem a rede GHENeSys.....47

Fig 4.2 Um exemplo de rede GHENeSys.....48

Fig 4.3 Uma rede com elemento "macro".....64

Fig. 4.4 Resultado do refinamento do elemento b_3.....64

Fig. 4.5 Diagrama das classes *Place* e *Activity*.....68

Fig 5.1 Modelagem do processamento de reclamações usando redes de Petri.....91

Fig. 5.2 Modelagem do processamento de reclamações no seu nível mais abstrato com a rede GHENeSys.....92

Fig. 5.3 Primeiro refinamento do modelo do processamento de reclamações.....93

Fig. 5.4 Versão final do modelo do processamento de reclamações.....93

Fig 6.1 Esquema do andar.....101

Fig. 6.2. Arquitetura baseada em íntegrons do sistema.....105

Fig 6.3 Modelo do integron do controle da sala.....108

Fig. 6.4 Modelo do controle de nível superior.....111

Fig. 6.5 Subrede do elemento macro **Processar informação** em sua versão cíclica.....113

LISTA DE TABELAS

Tabela 6.1 Estrutura do objeto *Marca*.....106

Tabela 6.2 Estrutura do objeto *Marca2*.....107

Tabela 6.3 Breve descripção dos métodos usados no controle da sala.....109

Tabela 6.3 Breve descripção dos métodos usados no controle da sala (continuação)..110

Tabela 6.4 Descrição dos métodos do modelo do nível superior de controle.....112

LISTA DE ABREVIATURAS

PFS/MFG	Production Flow Schema/Mark Flow Graph
ODRP	Objetos dentro de uma rede de Petri
RPDO	Redes de Petri dentro de objetos
DEMON	Design Methods Based on Petri Nets
AOO	Análise Orientada a Objetos
MFG	(Mark Flow Graph
E-MFG	Extended Mark Flow Graph
Pr/T	Predicado/Transição
P/T	Lugar/Transição
FIFO	First In First Out
FILO	First In Last Out
SFM	Sistemas Flexíveis de Manufatura
WFMC	Workflow Management Coalition
SDVC	Sistema Dinâmico de Variáveis Continuas
SDED	Sistemas Dinâmicos de Eventos Discretos

LISTA DE SÍMBOLOS

Símbolos lógicos e matemáticos

$\mathbb{N}$	Conjunto dos números naturais
$\mathbb{N}^+$	Conjunto dos números naturais sem o zero
$+$	Adição
$-$	Subtração
$*$	Multiplicação escalar
$\mid$	Tal que
$\wedge$	"e" lógico
$\vee$	"ou" lógico
$\cup$	União entre conjuntos
$\cap$	Interseção entre conjuntos
$\setminus$	Diferença entre conjuntos
$\varnothing$	Conjunto vazio
$\neq$	Diferente
$\geq$	Maior ou igual que
$\leq$	Menor ou igual que

$\notin$ Não pertence

$\in$ Pertence

$\subseteq$ Contido

$=$ Igual

$\forall$ Qualquer

$\exists$ Existe

$\sum$ Somatória

$\rightarrow$ Implica

∞ Infinito

Demais símbolos

M Uma marcação qualquer

R(**M**) Conjunto das marcações alcançáveis a partir de **M**

C Matriz de incidência

$\mathbf{C}^T$ Transposta da matriz **C**

$x^\bullet$ Pós-elementos do elemento x

${}^\bullet x$ Pré-elementos do elemento x

a_i i-ésimo elemento do vetor a

n, m, i, j Números naturais

σ Seqüências de disparo

σ^* Seqüência de disparo repetida várias vezes.

ℓ Um lugar qualquer.

L(**M**) Conjunto de todas as possíveis seqüências de disparo a partir de **M**

M[a>**M**$_1$ Ocorrência do evento a leva da marcação **M** a **M**$_1$

$\mathcal{M}$, **N**, **Nr** Uma rede

C_{mn} Elemento da fila m, coluna n da matriz C

x_m m-ésimo elemento do vetor x

A, **L**, **B**, **E**, **F**, **P** Conjuntos

A, **B**, **C** Matrizes

K Vetor coluna das capacidades

K(s) Capacidade do elemento s

ν_k Vetor de habilitação (vetor coluna)

M(l) Marcação do elemento l

RESUMO

Existe na comunidade de redes de Petri duas correntes distintas : i) uma que advoga o uso das redes de Petri como uma teoria, no sentido matemático do termo, deduzindo a análise do sistema (representado por um modelo formal), através de suas propriedades, que por sua vez são derivadas dos princípios e definições da rede; ii) a segunda corrente se preocupa mais com o uso prático das redes e está muito mais atenta à correspondência entre o modelo (mesmo parcialmente formal) e sua interpretação física, baseando o seu trabalho (geralmente) em extensões das redes.

Portanto, o que deveria ser um desenvolvimento sinérgico com contribuições mútuas de ambos os lados parece atualmente um divisor de águas na comunidade acadêmica.

Este trabalho pretende contribuir no sentido de aproximar estas duas tendências e mostrar que, de um lado, as extensões são na verdade sub-classes das Redes de Petri elementares onde boa parte dos atributos que aproximam cada uma destas extensões da aplicação podem ser encapsulados em métodos. Isto se utilizarmos uma versão orientada a objetos para as redes de Petri, onde cada elemento é um objeto, e a rede é, ao mesmo tempo, um agregado de objetos e uma sub-rede de um objeto gerador. Este esquema geral, que neste trabalho é apresentado apenas como uma superclasse de um conjunto de extensões, foi chamado de GHENeSys (General Hierarchical Enhanced Net System).

Neste trabalho, apresentamos a definição do GHENeSys na forma de rede, sua complementariedade com as redes de Petri convencionais, e sua capacidade de encapsular como sub-classes algumas extensões, através do mapeamento das funcionalidades destas com métodos pertencentes aos objetos ativos e passivos. Outro aspecto importante apresentado é a propagação das propriedades canônicas das redes na estrutura hierárquica do GHENeSys.

Uma comparação é feita com outras propostas de redes por objetos apresentadas no *1st Workshop on Object-Oriented Programming and Models of Concurrency* realizado em Lisboa em 1955, e com outras propostas mais recentes.

A aplicabilidade do GHENeSys é apresentada, embora tenhamos ainda um modelo preliminar, em problemas pertinentes à área de manufatura, como na análise de *workflow*, e em um estudo de caso que enfoca um problema de controle em malha fechada para sistemas prediais, enfatizando a integração do algorítmo de controle e a flexibilidade para a inclusão de políticas de acesso e uso das dependências do prédio, além da eliminação de sensores redundantes.

ABSTRACT

There are two general lines of work concerning the theory and practice of Petri Nets : i) one that treats Petri Net as a sound theory, based on first principles, and reducing the practice of Petri Net to the analysis of properties derived from the theory; ii) other, more concerned with an interpretation that can map a net model and practical issues of the target system, and that generally uses extensions to the conventional Petri Nets.

Thus, what should be a cooperative research agenda getting together theory and practice is in fact a polemic subject about which is the cornerstone of the research development in this area.

The intended contribution addresses a possible fusion between these two lines of approach to Petri Nets. To our object-oriented perspective of Petri Nets, extensions are in fact sub-classes of a general Elementary Net, where practical issues are represented by attributes and methods associated to each element. The overall net is an aggregate of these elements, and at the same time a subnet associated to one of then (emphasizing a structured approach). Such schema was called GHENeSys (General Hierarchical Enhanced Net System) which basic definitions are presented in this thesis.

In this work we present a net representation of GHENeSys, as well as its complementarity with the conventional net approach and its capacity – throughout a mapping of functionalities – to encompass extensions as representational subclasses. Also, we show that the classical properties of nets are preserved in the hierarchical structure of GHENeSys.

A comparison is made with similar proposals that appeared in the no *1st Workshop on Object-Oriented Programming and Models of Concurrency* and others presented more recently.

The suitability of GHENeSys to practical applications is showed, first in the workflow analysis, a typical manufacturing problem, and in a case study of closed loop control and supervision connected to intelligent building facilities. Integration and flexibility are an additional challenge to this application, where policies for the use of dependencies are set and changed according the demand. We claim that the observed redundancy of sensors that occurs in this kind of application can be eliminated with our approach.

1 INTRODUÇÃO.

No processo de *design* de sistemas complexos e integrados [Silva, 92] o grau de complexidade aumenta proporcionalmente a diferentes fatores: o tamanho do sistema, o grau de integração entre as partes que o formam ou a combinação de ambas, o que é muito freqüente. Este aumento de complexidade dificulta o processo de *design* de diversas formas. A solução do problema torna-se mais difícil, e por outro lado, o processo torna-se mais demorado aumentando também a possibilidade de erros, o que acaba por incidir no custo do projeto.

Existe uma busca na comunidade científica e acadêmica de métodos formais para a modelagem e design de sistemas. As vantagens dos métodos formais no processo de *design* são por exemplo, reutilização de módulos, repetibilidade da solução, a facilidade para proceder à análise de sistemas de grande porte, a precisão do processo de modelagem, a facilidade de documentação. Tudo isto justifica a obtenção de novos métodos que tornem mais rápido e seguro este processo.

Neste campo, as redes de Petri e as suas extensões, e a orientação a objetos, estão presentes de forma conjunta em muitas das propostas apresentadas na literatura. Isto deve-se principalmente ao fato da rede de Petri ter sido a representação/formalismo que mais

avançou no sentido de obter um bom balanço entre *soundness* e aplicabilidade, expressividade, e aderência aos métodos de *design* convencionais como o *top down* ou *bottom up*. Por outro lado, ficou ainda em aberto a possibilidade de se incluir neste *framework* alguns conceitos que se desenvolveram no campo da Ciência da Computação e que, por analogia, poderiam ser de grande utilidade no processo de modelagem e análise de sistemas integrados, tais como : reutilização, encapsulamento, modularidade e estruturação.

A proposta de uma rede de Petri juntamente com os conceitos da orientação a objetos pode oferecer os seguintes benefícios:

- um modelo simples de concorrência e sincronização.
- semântica operacional que suporte simulação e execução.
- técnicas de análises.
- abstração.
- refinamento.
- encapsulamento.
- reutilização e compartilhamento.

Ainda que muitos pesquisadores concordem com as vantagens da fusão da representação em redes de Petri com os conceitos de orientação a objeto, existem diferentes visões de como estes conceitos podem ser usados em conjunto [Moldt, 95].

Neste trabalho apresenta-se uma proposta de rede estendida que, em conjunto com um método para a construção da rede, proporcionam uma ferramenta poderosa para o projeto de sistemas complexos, integrados, e flexíveis que possam ser satisfatoriamente representados como sistemas a eventos discretos e isto vai depender do nível de detalhamento do sistema [Valette, 90]. Abordaremos a aplicação destes conceitos no gerenciamento de *workflows* e trataremos de estendê-los para outras aplicações não menos interessantes como são outros tipos de sistemas de controle e a própria programação.

Para dar suporte ao processo de *design* serão utilizadas as técnicas de análise baseadas no cálculo das propriedades dinâmicas e estruturais das redes de Petri, o que torna a manutenção da parte formal destas redes na versão orientada a objetos, fundamental para nossa proposta. A introdução do conceito de hierarquia e estruturação nesta proposta, ao lado do cálculo de propriedades, tais como invariantes e distância síncrona, facilitam a verificação das propriedades desejadas para o sistema nos diferentes níveis de abstração, garantindo um comportamento adequado e seguro (livre de bloqueios) além de facilitar a localização de possíveis falhas.

1.1 Motivação.

Fruto do desenvolvimento do setor industrial e de prestação de serviços - e principalmente da inversão do fluxo de demanda - que coloca o usuário e suas necessidades como uma parte fundamental no processo de *design*, muitas empresas foram obrigadas a modificar seus conceitos, principalmente nas áreas de projeto e produção. Em conseqüência

destas mudanças, da necessidade de subsistir e enfrentar a concorrência, estas empresas têm tido que investir na busca de meios para agilizar o *design* e produção de seus produtos, adaptando-se rapidamente às mudanças do mercado. Tudo isto fez com que aparecessem novos conceitos tais como sistemas de informação e *workflow* e com isso novos problemas de *design* – do fluxo de informação no chão de fábrica, no primeiro caso, e da alocação de recursos e tomadas de decisão no segundo caso.

Nos últimos anos, vários métodos têm sido propostos que reclamam para si um conteúdo formal aliado à facilidade de ser aplicado em um ambiente industrial, com uma boa expressividade, e finalmente, com uma grande aderência a métodos novos e/ou já conhecidos de *design* de sistemas. Estas características são antagônicas, e se um dado método se destaca em uma delas certamente paga um pesado tributo em algumas das outras.

Algumas das propostas nesta área encontradas na literatura abordam de algum modo a estruturação das redes, dada as vantagens que esta característica representa no projeto de sistemas complexos, porém não é mencionado como obter ou construir estas redes estruturadas, nem como validar uma rede estruturada, uma vez obtida.

O objetivo principal deste trabalho é apresentar regras e definições que permitam construir redes estruturadas para serem usadas em vários tipos de aplicações. Estas definições e regras estarão baseado na definição de programa estruturado e no teorema da estruturação de Bohn e Jacopini [Linger, 79], e usados no gerenciamento de *workflows,* entre outras

aplicações que possam ser representados através de redes de Petri estendidas.

Neste trabalho será apresentada uma proposta de rede estendida com características de hierarquia, orientação a objetos e estruturação inspirada o esquema PFS/MFG proposto por Miyagi [Miyagi, 88] e colocado na forma de rede por Silva e Miyagi [Silva & Miyagi, 95]. Esta rede resultará a base para a implementação de um sistema para construir modelos de forma estruturada, validá-los fazendo uso das propriedades das redes de Petri assim como da simulação, e finalmente apoiar a implementação dos sistemas modelados. O sistema será chamado de GHENeSys (**G**eneral **H**ierarchical **E**nhanced **Ne**t **Sys**tem) e o mesmo nome sera adotado neste trabalho para a rede proposta.

Além disso, será demostrado como, observando algumas normas para a construção e execução das redes, é possível utilizar os conceitos de orientação a objetos para aumentar o poder de representação da rede, entre outros benefícios, tais como encapsulamento e reutilização.

1.2 Organização do trabalho.

No Capítulo 2 apresenta-se uma revisão bibliográfica sobre redes de Petri orientadas a objetos, algumas propostas de estruturação de redes assim como de aplicações de redes na modelagem, análise e projeto de sistemas a eventos discretos. No Capítulo 3 apresenta-se as teorias e modelos que são a base deste trabalho. Neste capítulo são tratados os temas de estruturação, orientação a objetos, e definições sobre redes elementares, a serem usados na

definição da rede GHENeSys.

A partir do Capítulo 4 apresentam-se os elementos básicos da rede GHENeSys, a representação dos métodos e atributos dentro destes elementos, a equação de estado da rede, e as operações entre objetos que podem ser realizadas. No Capítulo 5 fazemos uma discussão sobre o processo de modelagem utilizando redes de Petri e do tipo de aplicações que podem ser modeladas garantindo estruturação, com o método proposto. Também são mostradas as vantagens desta forma de construir redes e algumas das interpretações possíveis das propriedades das redes que podem ser calculadas para apoiar o processo de *design*.

No Capítulo 6 é feito um estudo de caso aplicando a rede GHENeSys ao projeto de sistemas a partir das especificações, e apresentamos os resultados obtidos neste processo.

Finalmente no Capítulo 7 apresentam-se as conclusões deste trabalho assim como uma proposta de trabalho futuros.

2 REVISÃO BIBLIOGRÁFICA.

Existe uma grande variedade de trabalhos onde as redes de Petri são utilizadas para modelar e/ou apoiar o processo de análise e *design* de sistemas. Neste trabalho, pretendemos definir uma rede que, pelas suas características, possa ser utilizada em diferentes campos de aplicação tais como modelagem de sistemas de controle discreto e processos de negócios assim como sistemas de informação e fluxo de materiais.

Algumas abordagens são freqüentemente utilizadas nestes trabalhos, porém de forma diferente. Devido a potencialidade de algumas destas abordagens neste campo, considera-se conveniente analisar como estas vêm sendo usadas por diferentes pesquisadores que trabalham na área de modelagem e *design* com redes de Petri.

Neste capítulo pretendemos fazer um pequeno resumo das principais abordagens apresentadas nas publicações mais recentes nesta área.

2.1 Abordagens de Orientação a Objetos em redes de Petri.

Na última década tem aparecido na literatura trabalhos de pesquisa relacionando as redes de Petri com o paradigma de orientação a objetos. Isto é devido fundamentalmente às características das redes de Petri, tais como: uma representação gráfica, que permite modelar concorrência e sincronismo; um esquema de representação que, devido a sua natureza abstrata, pode ser usado em diferentes aplicações; um formalismo que permite calcular propriedades dinâmicas e estruturais aplicadas na análise do comportamento dos sistemas. Entretanto a ausência de modularidade, e de mecanismos de estruturação e encapsulamento constituem uma limitação das redes de Petri [Esser, 97]. Por outro lado, a orientação a objetos é conhecida justamente por conceitos tais como modularidade, reutilização, herança, delegação, etc. Isto sugere que estas abordagens podem se complementar e evoluir em conjunto numa abordagem comum onde as vantagens das duas são aproveitadas convenientemente.

As propostas englobando estes dois paradigmas são muitas e diferentes entre si. Estas podem ser agrupadas em duas grandes tendências [Bastide, 95]:"Objetos dentro de uma rede de Petri" (ODRP) e "Redes de Petri dentro de objetos" (RPDO). Justamente a partir do ano 95 começaram a aparecer propostas de unificação destas duas tendências.

A seguir faremos uma pequena discussão de algumas abordagens das diferentes tendências.

2.1.1 Objetos dentro de uma rede de Petri.

Nestas abordagens os elementos da rede são vistos como objetos, em alguns casos as marcas são descritas como operações algébricas e em outras são descritas como objetos passivos (não contém métodos). A rede representa a estrutura de controle do sistema, enquanto as marcas modelam a estrutura de dados do sistema.

Dependendo do contexto a semântica da ocorrência das transições pode ser vista de forma diferente. Para alguns autores, os objetos "marca" não são criados e destruídos cada vez que uma transição é disparada. Nestes casos se considera que a marca move-se de um lugar a outro usando métodos que pertencem aos objetos "lugares" envolvidos na transferencia. Evidentemente, uma transição pode também instanciar um novo objeto ou destruir um existente. Nesta categoria estão os trabalhos apresentados em [Battiston, 90] e [Lakos, 94].

2.1.2 Redes de Petri dentro de objetos.

Nesta abordagem as redes são usadas para modelar o comportamento de processos internos aos objetos. Deste modo a rede modela a disponibilidade dos métodos de um objeto, permitindo a execução concorrente dos mesmos. As redes também são usadas para representar a comunicação entre objetos [Bastide, 95].

Neste caso os objetos são utilizados para inserir o conceito de modularidade no sistema sendo modelado, mais do que introduzir o conceito de estruturação e hierarquia. Outra funcionalidade da abordagem é modelar objetos (ou módulos funcionais) que tenham comportamento interno concorrente (processos de usinagem/ movimentação de peças

dentro de ilhas de automação, centros de usinagem, etc.).

2.1.3 Unificação das tendências.

A unificação da abordagem ODRP e RPDO deve enfrentar basicamente as seguintes barreiras conceituais:

- achar uma classe mãe única, tanto para os elementos estáticos (lugares), quanto para os elementos dinâmicos (atividades) (uma contraproposta seria admitir uma classe mãe genérica T que teria duas subclasses).

- resolver se as marcas seriam ou não uma classe (uma terceira classe talvez) e o seu significado, o que resultaria em uma rede de alto nível.

- em havendo o conceito de hierarquia, resolver, dentro do conceito de objetos como introduzir a hierarquia em harmonia com o conceito de modularidade e encapsulamento.

- resolver a introdução de herança múltipla e relações especiais entre objetos como a delegação (*delegation*).

- resolver como implementar o conceito de cooperação (e transferência de mensagens) entre os elementos sem mutilar o formalismo de redes.

- definir a dinâmica de criação dos objetos (objetos podem ser criados e destruídos dinamicamente durante o ciclo de vida dos sistemas, notadamente as marcas).

- definir o relacionamento entre objetos, no caso em que as marcas são também consideradas como tal (abordagem de alto nível). Neste caso, se a marca é considerada uma referência para outras sub-redes, a sucessiva ocorrência de uma mesma transição pode levar a diferentes referências.

- problemas para integrar o conceito de polimorfismo as redes.

Na proposta de unificação das tendências do próprio Bastide [Bastide, 95] são apontados alguns destes problemas.

Uma outra proposta unificadora encontrada na literatura, visando resolver o problema da diversidade de superclasses admite que subredes sejam colocadas nas marcas. Desta forma a diferenciação de superclasses (uma para as redes e outra para as marcas) não é mais necessária e é usada uma única hierarquia de classes [Lakos, 95].

Os objetos cooperativos apresentados por Sibertin-Blanc em [Sibertin-Blanc, 98] apresentam uma abordagem abstrata para a fusão entre objetos e redes de Petri, mas impõem, ao mesmo tempo, uma disciplina de projeto adequada para sistemas concorrentes e que segue a linha dos *actors* propostos por [Agha, 86]. Se por um lado os objetos cooperativos atendem à necessidade de abrangência e generalidade que queremos atribuir às redes de Petri, por outro lado se afastam da modelagem de sistemas discretos integrados e flexíveis e também da interpretação das redes e seus elementos, que seria importante explicitar mais claramente.

Entretanto, a abordagem de objetos cooperativos, além de aproximar a modelagem do conceito de agentes – o que resulta interessante para uma abordagem de controle discreto, - contextualiza cada objeto em um contexto, através de uma denominada "interface". Assim, cada objeto é de fato um sistema independente e é concebido em um espaço de quatro dimensões [Sibertin-Blanc, 98] : a das entidades processadas pelo sistema, a das operações que constituem o processo, a dos atores que executam as operações, e a da estrutura de controle que define o seu comportamento.

Em contrapartida ao seu caráter abstrato, Sibertin-Blanc desenvolveu um compilador chamado SYROCO para transformar esquemas de objetos cooperativos em código C++ [Sibertin-Blanc, 96], amenizando os problemas de implementação. No entanto o formalismo das redes de Petri, associado à modelagem de sistemas dinâmicos discretos pelo paradigma estado-transição[1] não pode ser mantido, dando lugar a uma nova disciplina de projeto baseada nos objetos cooperativos. Esta nova disciplina se encaixaria adequadamente para sistemas que admitem uma arquitetura realmente distribuída. Se entretanto admitimos que existem, para a arquitetura de sistema de manufatura discreta, outras possibilidades, tais como a hierárquica e hierárquica modificada [Dilts, 91], estas seriam modeladas com bastante dificuldade pêlos objetos cooperativos.

Em [Lakos, 95], apresenta-se uma abordagem similar a anterior, mas desta vez a rede orientada a objeto conserva um enlace com as redes coloridas propostas por Jensen. [Jensen

---

[1] Devido ao caracter bi-partido das redes de Petri.

96]. O objetivo do autor é aumentar o grau de integração dos conceitos de orientação a objetos dentro da teoria das redes de Petri. Foi desenvolvida uma extensão das redes de Petri coloridas incluindo canais de sincronização e extensões nos arcos com as suas respectivas equivalências nas redes de Petri coloridas. Estes elementos adicionais foram criados para permitir a comunicação entre os diferentes objetos ou classes e para garantir a conexão entre diferentes objetos é níveis de abstração. É através deste mecanismo que a hierarquia é definida nesta linguagem. A limitação desta abordagem está na complexidade do processo para a obtenção do modelo orientado a objetos, e neste sentido a proposta é usar os métodos apresentados em [Booch, 91], [Jacobson, 92] e [Rumbaugh, 91] para definir as classes, objetos e inter-relações entre estes. Devido ao enlace desta linguagem com as redes coloridas a verificação das propriedades da rede está restrita ao cálculo de invariantes. Além disso, esta verificação só é possível uma vez concluído o modelo.

A abordagem apresentada em [Esser, 97] é similar a apresentada em [Lakos, 95] exceto que, na primeira, a rede foi estendida com os conceitos de tempo e de instanciação do sistema. Para a interconexão dos diferentes módulos ou superclasses foi criado o elemento interface que serve para declarar as conexões de uma certa superclasse com o resto das classes ou superclasses. Contudo, não é possível testar os modelos estando estes parcialmente construídos, portanto não podem ser detectados erros através do cálculo das propriedades das redes durante o processo de criação do modelo. Apesar desta proposta ser similar a de Lakos o enlace com as redes de Petri coloridas não é conservado. Esta abordagem não apresenta nenhuma proposta para resolver o problema de estruturação das

redes.

2.1.4 Resumo das abordagens anteriores.

Na maioria dos casos, as abordagens orientadas a objetos para as redes de Petri estudadas até aqui, são colocadas em algum tipo de linguagem orientada a objetos, portanto o uso destas ferramentas para projetar sistemas (principalmente quando estes não são desenvolvidos para serem programas de computador) não possuem uma base formal que permita a validação dos modelos construídos.

Outro aspecto para levar em consideração é que as Redes de Petri orientadas a objetos ainda é um conceito não muito claro pois, várias abordagens diferentes são chamadas assim. Um dos maiores problemas é que a maioria das definições só servem para áreas de aplicações específicas, e não foram desenvolvidas para aplicações gerais no campo da especificação de sistemas [Moldt, 95].

Quanto ao formalismo, as abordagens apresentadas na literatura não apresentam um formalismo "sound" para objetos (pelo menos não um que seja consenso na comunidade acadêmica), o que dificulta muito a obtenção de uma representação que expresse a fusão com o formalismo de redes de Petri. As soluções apresentadas ganham em poder de representação (representam os sistemas de forma sintetizada o que contribui para minimizar o problema da explosão de estados) e em estruturação, mas perdem no que, ao nosso ver, é uma grande arma das redes de Petri para a análise de sistemas discretos: o cálculo das propriedades e a validação dos modelos.

Mesmo nos casos onde ainda é possível validar o modelo resultante, os processos de validação só se aplicam no nível de abstração mais baixo quando toda a modelagem já foi concluída, o que no mínimo está em desacordo com o fato de que os problemas de *design* são pertinentes às fases iniciais de concepção e modelagem, sendo também a fase onde a correção de erros custa menos [Sihng, 98][Silva, 92].

2.2 Abordagens para a estruturação das redes de Petri.

Devido às vantagens dos modelos estruturados, a procura por métodos para construir e validar estes modelos tem se tornado muito atraente na comunidade acadêmica. Note-se entretanto que as redes de Petri não são adequadas para modelar um sistema com um conjunto de componentes interativos, uma vez que o modelo resultante deste processo não é estruturado [Sibertin-Blanc, 98]. Este problema pode ser minorado se forem colocadas restrições na rede e/ou se forem definidos métodos para sintetizar modelos estruturados (estruturados por construção).

Na literatura, este problema é abordado utilizando noções diferentes do conceito de estruturação, de acordo com o problema específico a que é destinado. A maioria dos trabalhos é direcionado ao estudo das estruturas das redes de Petri, entendendo como estrutura o conjunto de elementos básicos que fazem parte da rede e a interconexão entre eles. Nos outros casos o conceito de estrutura utilizado é o mesmo utilizado na

programação estruturada, procurando elementos próprios e primos (sub-redes irredutíveis) para a formação de uma rede mais elaborada.

2.2.1 O Conceito de estrutura nas redes de Petri.

Em [Best, 87] uma revisão dos principais aspectos e contribuições para o conceito de estrutura é apresentado. Os conceitos de estrutura e comportamento nas redes de Petri utilizados são os seguintes: a estrutura de uma rede de Petri é definida pela enupla (B, E; F) onde B são os lugares, E as transições e F os arcos que interconectam estes elementos; e pelas propriedades que são independentes da marcação inicial; o comportamento de uma rede de Petri é dado pelas propriedades dependentes da marcação (dinâmicas), isto é, as propriedades relacionadas à evolução das marcas segundo uma regra de disparo da rede (considerando todas as possíveis combinações, na evolução dos estados).

Definidos desta forma, podemos assegurar que o comportamento da rede é a parte do modelo mais aderente ao modelo físico do processo (análise das relações causa-efeito), sendo este derivado de um sistema dinâmico discreto. Normalmente, o comportamento de uma determinada rede é mais fácil de analisar que sua estrutura, dada a natureza combinatória desta última. Alguns exemplos de propriedades para a análise do comportamento das redes são a vivacidade, segurança, existência de invariantes, distância síncrona.

O estudo da estrutura nas redes de Petri procura investigar as relações que possam existir entre o comportamento de uma rede marcada e a estrutura desta mesma rede sem marcas.

Existem várias publicações nesta área que mostram resultados neste sentido [Esparza, 90].

O objetivo da procura destas relações, é o de utilizar as mesmas para analisar os sistemas modelados através de redes, focando nas propriedades estruturais uma vez que estas são mais gerais e devem se verificar independentemente da marcação inicial escolhida. O raciocínio anterior é uma das bases que utilizaremos neste trabalho e que resultará de grande ajuda para procurar estruturas de redes com propriedades "desejáveis" e com grande poder de representação.

Alguns resultados reportados na literatura neste sentido serão apresentados a medida que sejam utilizados neste trabalho.

2.2.2 As redes de Petri estruturadas.

Recentemente, no trabalho de W. M. P. Van der Alst sobre aplicação das redes de Petri ao gerenciamento de processos de negócios, foram introduzidas as redes de fluxos de trabalho (Workflow Nets) [van der Aalst, 97]. Este tipo de rede garante, por construção, os requerimentos mínimos para a não existência de alguns tipos de bloqueios comuns no processo de modelagem. Considerasse que isto é um primeiro passo para chegar as redes de Petri estruturadas.

Segundo a norma da WorkFlow Management Coalition, os roteamentos dos processos de negócios a serem modelados são os roteamentos seqüenciais, condicionais, paralelos, e iterativos [WFMC, 96]. Blocos para a construção de cada um destes roteamentos estão bem explicitados em [van der Aalst, 98]. Fazendo uma analogia com o teorema da

estruturação de Bohn e Jacopini [Linger, 79], estes blocos garantem a construção de um esquema devidamente estruturado, pois também formam uma base de blocos primos e próprios capaz de gerar qualquer estrutura de controle ou *workflow*.

Esparza e Silva definiram estruturas que foram denominadas de "*handles*" e "*bridges*", as quais foram introduzidas para complementar os bloqueios [Esparza & Silva, 90]. Além disso desenvolveram métodos de sínteses de redes livres de escolha (*free choice nets*) através da sincronização de máquinas de estados.

Além de pesquisar as "estruturas" de redes que garantam um comportamento adequado, Esparza e Silva procuram identificar condições suficientes e/ou necessárias para obter um "bom" comportamento. Estes aspectos são tratados nos artigos anteriormente citados e utilizam como base para estes estudos as redes livres de escolha. Este tipo de redes é muito usada nestes estudos devido ao seu poder de representação para processos de negócios e que possuem um grande número de resultados reportados na literatura sobre a análise das suas propriedades [Best, 87], [Desel & Esparza, 95].

2.3 O uso das redes de Petri na modelagem, análise e projeto de sistemas.

As redes de Petri possuem um grande potencial para a modelagem de sistemas com concorrência, e sistemas a eventos discretos entre outros. Este potencial é baseado em três aspectos fundamentais que compõem esta teoria: uma noção básica de não determinismo,

uma noção básica de concorrência e uma noção básica de seqüência [Best, 90]. Com estas três noções básicas é possível obter uma grande variedade de outros conceitos e propriedades. Outra motivação é o fato dos modelos em redes de Petri adaptarem-se facilmente a definições de diferentes semânticas comportamentais, tais como intercalada, de passo e de ordem parcial entre outras.

Muitos trabalhos em diferentes áreas utilizam as redes de Petri como abordagem principal para representar os sistemas. Uma visão da aplicação das redes de Petri na automação e na informática pode ser encontrada em diferentes publicações [Reisig, 82][Silva M, 85], [Murata, 89][Cao, 90].

Apesar das redes de Petri serem consideradas uma excelente ferramenta para a modelagem, esta não constitui a única razão para o seu uso cada vez mais generalizado nos diferentes ramos da engenharia e da informática. A existência de um grupo de propriedades interessantes que podem ser intuitivamente e satisfatoriamente expressas em redes de Petri, tais como a vivacidade, a equidade, etc., fazem com que esta ferramenta possa ser aplicada também à análise de sistemas e portanto também ao seu projeto.

Neste sentido, também podem ser encontradas publicações que podem ser separadas em dois grandes grupos: as que tratam de propriedades de redes de Petri, sua interpretação física e dos métodos para calcula-las, e as que desenvolvem métodos de análise para determinadas aplicações baseadas em algumas propriedades de redes (as propriedades mais importantes para estas aplicações). No primeiro grupo podemos citar as seguintes

publicações: [Goltz, 86], [Desel, 92], [Murata ,89], [Esparza 00]. No segundo grupo podemos encontrar publicações tais como [Prock, 92], [Colom, 90], [Esparza, 92].

Outro aspecto de interesse abordado na última década tem sido os métodos de projeto baseados em redes. Em [Best, 90] foram colocadas algumas das razões fundamentais para a utilização das redes de Petri no projeto de sistemas e foram relacionadas as atividades e diferentes grupos temáticos associados ao projeto DEMON (Design Methods Based on Petri Nets), no escopo do ESPRIT. Neste projeto foram identificados três temas básicos: o desenvolvimento de classes de redes de Petri unido a conceitos de estruturação; definição, classificação e avaliação de noções de simulação; e a pesquisa de técnicas para provar e verificar propriedades de redes.

Neste contexto foi publicado o trabalho de Brauer, Gold e Vogler [Brauer, 90] onde um resumo sobre a preservação de propriedades em componentes refinados em redes de Petri e equivalência (funcional) entre redes é apresentado, fazendo referência a trabalhos anteriores de Valette [Valette, 79] e Suzuki [Suzuki, 83]. Estes pontos são fundamentais quando se fala de "*design top-down*" de sistemas com concorrência.

Outros trabalhos foram feitos neste mesmo período onde as redes de Petri foram aplicadas à análise e sínteses de sistemas. Esparza e Silva fizeram grandes contribuições nesta área introduzindo as "regras de redução" para síntese, e técnicas para caraterizar algumas propriedades de redes em termos algébricos, para apoiar a análise, tomando como base as redes livres de escolha [Esparza & Silva, 90].

3 TEORIAS E MODELOS BÁSICOS.

Neste capítulo apresentaremos teorias e abordagens que servem como base para este trabalho.

3.1 Estruturação em programação.

O conceito de estruturação foi proposto para melhorar a organização dos programas maiores e mais complexos, e assim conseguir diminuir os custos de desenvolvimento e manutenção. Por causa disto, foi no âmbito da ciência da computação que a estruturação se desenvolveu, passando por conceitos como programas próprios, programas primos até chegar ao teorema da estruturação.

A seguir apresentaremos alguns destes conceitos que são básicos para se entender a estruturação em linguagens de programação.

Um programa próprio é um programa com uma estrutura de controle que:

1. Tem uma única entrada e uma única saída, e

2. Para cada nó (junção de um ou mais fluxos de controle), existe um caminho

entre a entrada e a saída que passa através deste nó.

Um programa próprio pode conter partes que por sua vez são programas próprios, estas partes podem ser chamadas de sub-programas próprios.

Um programa primo é um programa próprio que não tem sub-programas próprios com mais de um nó.

As estruturas de controle normalmente usadas em linguagens de programação são exemplos de programas primos: seqüência, se ..então, repetir .. até, fazer ...enquanto entre outras.

Um programa composto é um programa obtido a partir da agregação de programas primos.

Programas compostos podem ser construídos tão grandes quanto seja preciso. Quanto maior for um programa composto maior será a sua complexidade. Além do tamanho de um programa, a complexidade pode estar determinada pela base de programas primos utilizados para a construção de um determinado programa composto. Construir programas compostos com uma base adequada pode representar uma diminuição da complexidade do mesmo.

Um programa estruturado é um programa composto construído a partir de uma base fixa de programas primos.

Para linguagens de programação o conjunto básico formado pelas estruturas de controle conhecidas como se...então-senão (if-then-else), seqüência e enquanto ... fazer (while-do), constitui uma base sobre a qual se pode construir qualquer programa. Existem outras

estruturas de controle que são declaradas como base em linguagens de programação. Estas estruturas também podem ser utilizadas para montar uma base de programas primos. Porém, se a base canônica mostrada acima for utilizada, estruturas como repetir...até (repeat-until) podem ser geradas como uma combinação das estruturas básicas (seqüência, se...então-senão, enquanto...fazer).

O teorema da estruturação garante que qualquer programa próprio tem um programa estruturado equivalente construído a partir de um conjunto base de programas primos formado por { se...então senão, seqüência e enquanto ... fazer}.

3.2 Orientação a Objetos

A grande desvantagem da abordagem estruturada frente a orientação a objetos é a estrita separação das operações e dos dados. Já na abordagem orientada a objetos operações e dados são integrados de uma forma natural [Hubbers & Hofstede, 97]. Esta é uma das razões para que num espaço relativamente curto de tempo, inúmeros métodos orientados a objetos tenham sido propostos na literatura visando disciplinar o processo de *design* de sistemas. Consequentemente, existe uma grande diversidade de conceitos e definições que, muitas vezes, usam os mesmos nomes. Se adicionamos a isso tudo a falta de formalização de muitos destes métodos (e do próprio conceito de objeto) o resultado é uma grande dificuldade na compreensão precisa e no uso destes métodos.

A seguir apresentamos alguns conceitos básicos da abordagem orientada a objetos que

serão usados neste trabalho.

Fig. 3.1 Classe Pessoa e duas de suas instâncias: Maria e Pedro.

Uma classe é um conjunto de objetos similares. Uma instância de uma classe é chamada Objeto da Classe. Assim por exemplo, tem-se, na Figura 3.1, a Classe Pessoa, cujos atributos são Nome, Endereço, Sexo, Identidade, Estado Civil e outros, e cujos serviços são Estudar, Trabalhar, Dirigir, Passear, e outros. Na mesma figura tem-se que Maria e Pedro são duas instâncias da classe Pessoa, portanto são dois objetos dessa classe.

Abstração é a habilidade de ignorar os detalhes na caracterização de uma entidade que não sejam relevantes para o propósito em questão, salientando apenas as características principais [Oxford, 86]. Na modelagem de sistemas, normalmente é feita a seleção de alguns aspectos (ou propriedades) e supressão de outros, de forma que sejam representamos aquelas propriedades que são fundamentais para compreender cada componente do sistema

que estamos tentando representar. Os entes e conceitos do mundo real são normalmente complexos, portanto, para lidar com esta complexidade selecionamos parte do que estamos analisando, em vez de tentarmos compreender o sistema como um todo (método da divisão e conquista).

Num processo genérico de modelagem tenta-se inicialmente identificar as classes que poderão representar o sistema.

Atributo é qualquer propriedade, qualidade ou característica que pode ser atribuída a um ente ou objeto [Webster's, 77]. Na Análise Orientada a Objetos (AOO), o termo atributo é definido de forma a refletir o domínio do problema e as responsabilidades do sistema, ou seja, atributo é um dado (informação de estado) que tem um valor definido para cada objeto (instância) de uma classe.

Serviço (ou método) é uma atividade executada para permitir que as pessoas utilizem alguma coisa [Webster's, 77]. Na AOO, o termo serviço é definido de forma a refletir o domínio do problema e as responsabilidades do sistema, ou seja, serviço é um comportamento específico que um objeto deve exibir.

Muitas vezes pode-se ter uma classe que é quase, mas não exatamente o que queremos e, neste caso, a herança é uma técnica útil para ampliar a abstração de dados. Baseado nestes conceitos, podemos dizer que herança é o mecanismo para expressar a similaridade entre classes, reduzindo a definição de classes iguais a outras (mais abstratas) que já foram definidas (super-classe).

Este princípio permite representar membros comuns, serviços e atributos uma única vez, assim como especializar estes membros em casos específicos.

A herança permite, portanto, a reutilização de especificações comuns, logo no início das atividades de análise. Existem diferentes tipos de herança: simples e composta. Na herança simples uma classe pode herdar atributos e serviços de outra, na composta (múltipla) a classe compartilha a estrutura e o comportamento definidos em várias classes de forma seletiva. Neste trabalho utilizaremos o conceito de herança simples que também é conhecido como herança pura ou herança monotônica.

O reconhecimento da similaridade entre classes forma uma hierarquia de classes, onde superclasses representam abstrações generalizadas e subclasses representam refinamentos, onde atributos e serviços específicos são adicionados, modificados ou removidos. As classes são conectadas através de mecanismos de generalização e especialização, tornando explícitos os atributos e serviços comuns em uma hierarquia de classes.

Na figura 3.2, tem-se, por exemplo, que Estudante é uma Pessoa, Professor é uma Pessoa. Um objeto da classe Estudante, tem não apenas os atributos e serviços de Estudante, como também os atributos herdados da classe Pessoa. Por exemplo, o estudante João tem atributos e serviços de Pessoa, como Nome, Endereço, Dirigir, Viajar, e atributos e serviços que expressam o comportamento específico de um estudante, como Curso, Nota, Estudar e Realizar Prova. A classe Pessoa é reconhecida como uma generalização de Estudante, e este, por sua vez, como uma especialização, a qual herda os serviços e atributos de Pessoa.

Além disso, uma classe derivada (uma especialização) pode incluir novos serviços e

atributos ou redefinir os serviços herdados. Dessa forma, Funcionário pode incluir atributos como Salário e Profissão e serviços como **CalcularSalário**, que podem não ser relevantes ou apropriados para Pessoa em geral.

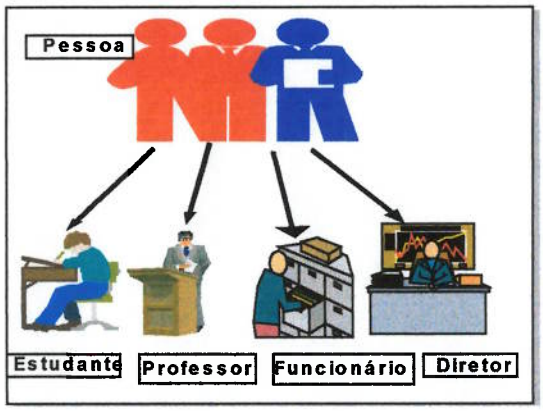


Fig. 3.2 Herança: Classe Pessoa e classes derivadas.

A agregação é um mecanismo que permite a construção de uma classe agregada a partir de outras classes componentes. Por exemplo, na Figura 3.3 tem-se que uma casa tem portas, janelas, paredes, escadas e outras partes.

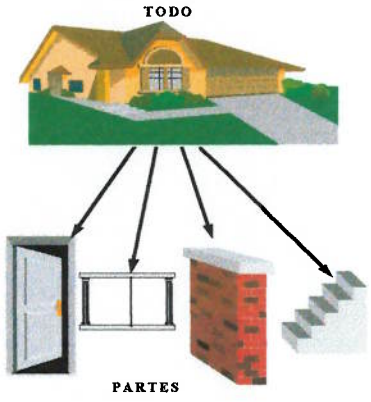


Fig. 3.3. Agregação: classe Casa e as classes que a compõem.

Mesmo em se tratando de entidades abstratas, pode-se compor um objeto a partir de outros objetos.

A associação vem do relacionamento entre as entidades do mundo real, e é usada para agrupar certos objetos que ocorrem em algum ponto no tempo ou sob circunstâncias similares.

Associação é uma união ou conexão de idéias [Webster's, 77]. Na AOO, a associação é modelada através de uma conexão de ocorrências. Uma conexão de ocorrência é um relacionamento que um objeto precisa ter com outro(s) objeto(s), para cumprir suas responsabilidades.

3.3 Definições básicas sobre redes de Petri e suas propriedades.

Derivada da Teoria de Grafos e da representação de autômatos finitos, as redes de Petri também se aplicam à otimização, análise e validação de sistemas, fornecendo portanto suporte às várias atividades essenciais na modelagem de sistemas dinâmicos discretos. Concretamente, a modelagem e análise se aplicam ao comportamento dinâmico do sistema e à análise de propriedades estruturais, a primeira baseada em simulação e/ou na resolução parcial da equação de estado e a segunda na derivação de propriedades para classes especiais de redes.

Em [Bernardinello, 92] são identificados três níveis diferentes de redes considerando os modelos de redes "básicos":

1) No primeiro nível estão as redes nas quais os lugares possuem no máximo uma marca.

2) No segundo nível estão as redes nas quais os lugares possuem várias marcas.

3) No terceiro nível estão as redes denominadas de "alto nível" quer dizer, as redes nas quais os lugares possuem várias marcas e estas por sua vez encapsulam informação.

Neste capítulo apresentamos algumas das propriedades básicas das redes de Petri. Logo a seguir, apresentaremos as redes estendidas e as redes de alto nível de forma resumida.

A seguir faremos una breve introdução destes modelos de redes usando a terminologia utilizada em [Reisig 82].

3.3.1 Definições dos modelos de redes.

Uma rede é uma tripla **N=(S, T; F)** onde:

i) $\mathbf{S} \cap \mathbf{T} = \varnothing$;

ii) $\mathbf{S} \cup \mathbf{T} \neq \varnothing$;

iii) $\mathbf{F} \subseteq (\mathbf{S}\mathbf{x}\mathbf{T}) \cup (\mathbf{T}\mathbf{x}\mathbf{S})$

iv) $dom(\mathbf{F}) \cup cod(\mathbf{F}) = \mathbf{S} \cup \mathbf{T}$

onde $dom(\mathbf{F}) = \{ x \in \mathbf{S} \cup \mathbf{T} \mid \exists y \in \mathbf{S} \cup \mathbf{T}. (x,y) \in \mathbf{F} \}$

$cod(\mathbf{F}) = \{ y \in \mathbf{S} \cup \mathbf{T} \mid \exists x \in \mathbf{S} \cup \mathbf{T}. (x,y) \in \mathbf{F} \}$

Para uma rede **N** onde $\mathbf{X} = \mathbf{S} \cup \mathbf{T}$ e $x \in \mathbf{X}$ temos as seguinte definições:

- Denominamos ${}^\bullet x = \{ y \in \mathbf{X} \mid (y,x) \in \mathbf{F} \}$ o conjunto dos pré-elementos de x.
- Denominamos $x^\bullet = \{ y \in \mathbf{X} \mid (x,y) \in \mathbf{F} \}$ o conjunto dos pós-elementos de x.
- Um par $(s,t) \in \mathbf{S} \times \mathbf{T}$ é chamado de laço se e somente se $(s,t) \in \mathbf{F}$ e $(t,s) \in \mathbf{F}$.
- Uma rede **N** é chamada de pura se e somente se **N** não contém laços.
- Uma rede **N** é chamada isolada se e somente se ${}^\bullet x \cup x^\bullet = \varnothing$.
- Uma rede **N** é chamada S-simples se e somente se $\forall s, s' \in \mathbf{S} : ({}^\bullet s = {}^\bullet s' \wedge s^\bullet = s'^\bullet) \Rightarrow s = s'$.
- Uma rede **N** é chamada T-simples se e somente se $\forall t, t' \in \mathbf{T} : ({}^\bullet t = {}^\bullet t' \wedge t^\bullet = t'^\bullet) \Rightarrow t = t'$.
- Uma rede **N** é chamada simples se e somente se **N** é T-simples e S-simples.

3.3.2 Redes Elementares.

Antes de definir as redes elementares devemos definir "*cases*" e passos.

Seja uma rede **N=(B ,E ;F)**

- O subconjunto $c \in \mathbf{B}$ é chamado de *case* e representa o conjunto de condições ativas na configuração.

- Seja $e \in \mathbf{E}$, $c \in \mathbf{B}$ dizemos que e está habilitado em c se e somente se ${}^\bullet e \subseteq c \wedge e^\bullet \cap c = \varnothing$.

- Seja $e \in \mathbf{E}$, $c \in \mathbf{B}$ com e habilitado em c. Então $c' = (c \setminus {}^\bullet e) \cup e^\bullet$ é chamado o resultado da ocorrência de e em c e se escreve: $c[e>c'$.

- Seja $u \subseteq \mathbf{E}$ dizemos que u é independente e denotamos como Ind(u), se e somente se $\forall e_1, e_2 \in u;\ e_1 \neq e_2 \Rightarrow ({}^\bullet e_1 \cup e_1{}^\bullet) \cap (e_2{}^\bullet \cup e_2{}^\bullet) = \varnothing$.

- Um passo é um conjunto de eventos $u \subseteq \mathbf{E}$ habilitados em c. Isto acontece se as seguintes condições são cumpridas:

  i) Ind(u)

  ii) ${}^\bullet u \subseteq c$

  iii) $u^\bullet \cap c = \varnothing$.

A representação gráfica das relações de fluxo são arcos que indicam o sentido do fluxo das marcas.

Uma sistema de redes elementares é uma quádrupla **N=(B,E;F,$\mathbf{c_0}$)** onde **(B,E;F)** é a rede subjacente a **N** e $\mathbf{c_0}$ é o case inicial de **N**.

As redes condição/evento e lugar/transição podem ser obtidas a partir de modificações básicas das redes elementares. No caso das redes condição/evento o case inicial da rede é substituído pelo conjunto de todas as marcações possíveis da rede. Nas redes lugar transição o número de marcas num mesmo lugar pode ser maior que um e a relação de fluxo também é modificada incluindo uma função de peso que indica o número de marcas que fluem através de cada arco.

A seguir apresentamos a definição de rede lugar/transição.

Uma rede lugar/transição é uma sêxtupla $\sum = (\mathbf{S},\mathbf{T};\mathbf{F},\mathbf{K},\mathbf{W},\mathbf{M_0})$ onde:

i) **(S, T; F)** é uma rede onde os elemento **S** são chamados de lugares, os elementos **T** são chamados de transições e os elementos de **F** são chamados de arcos.

ii) $\mathbf{K}: \mathbf{S} \rightarrow \mathbb{N}^+ \cup \{\infty\}$ é uma função de capacidade.

iii) $\mathbf{W}: \mathbf{F} \rightarrow \mathbb{N}^+$ é uma função de peso.

iv) $\mathbf{M_0}: \mathbf{S} \rightarrow \mathbb{N}$ é uma marcação inicial que satisfaz: $\forall s \in \mathbf{S}: M_0(s) \leq K(s)$.

Dada uma rede lugar/transição $\sum = (\mathbf{S},\mathbf{T};\mathbf{F},\mathbf{K},\mathbf{W},\mathbf{M_0})$:

i) A aplicação $\mathbf{M}: \mathbf{L} \rightarrow \mathbb{N}^+$ é chamada uma marcação em $\sum$ se e somente se $M(s) \leq K(s)$ para todo $s \in \mathbf{S}$, $\mathbf{L} \subseteq \wp(S)$.

ii) Seja **M** uma marcação em $\sum$ podemos dizer que uma transição $t \in \mathbf{T}$ está habilitada se e somente se:

$$\forall s \in \mathbf{S} : W(s,t) \leq M(s) \leq K(s) - W(t,s)$$

iii) Uma marcação **M** que habilita t leva a uma marcação **M'** disparando t onde

$$M'(s) = M(s) - W(s,t) + W(t,s) \quad \forall s \in \mathbf{S}$$

iv) A ocorrência de t leva de uma marcação **M** a **M'**, o que é representado por $\mathbf{M}[t>\mathbf{M'}$.

v) Denotamos por R(**M**) o conjunto das marcações de $\sum$ alcançáveis a partir de **M** tal que:

1) $\mathbf{M} \in R(\mathbf{M})$ e

2) se $\mathbf{M_1} \in R(\mathbf{M})$ e para algum $t \in \mathbf{T}$, $\mathbf{M_1}[t>\mathbf{M_2}$ então $\mathbf{M_2} \in R(\mathbf{M})$.

3.3.3 Algumas propriedades das Redes.

Nesta seção trataremos das propriedades de redes úteis para definir e validar o comportamento de sistemas modelados através das redes de Petri.

A vivacidade é uma propriedade que está relacionada com a ausência de bloqueios numa rede. Uma rede $(\mathbf{N},\mathbf{M_0})$ é dita viva se, independentemente da marcação alcançável a partir de $\mathbf{M_0}$, qualquer transição da rede é possível ser disparada partindo desta marcação através de uma determinada seqüência de disparos.

Mesmo sendo a vivacidade uma propriedade desejada em muitos sistemas, muitas vezes esta é uma restrição muito forte para ser verificada. Por este motivo Murata relaxou está definição dividindo esta propriedade em diferentes níveis de vivacidade [Murata, 89].

A definição destes níveis é a seguinte:

Dada uma rede de Petri $\mathbf{N=(S,T;F,K,W,M_0)}$, uma transição $t \in \mathbf{T}$ é dita:

0) morta (L0-viva), se *t* não aparece em nenhuma seqüência de disparo de $L(\mathbf{M_0})$.

1) Potencialmente disparável (L1-viva), se *t* aparece ao menos uma vez em alguma seqüência de disparo de $L(\mathbf{M_0})$.

2) L2 viva, se, dado um número $k \in \mathbb{N}^+$, *t* aparece ao menos k vezes em alguma seqüência de disparo de $L(\mathbf{M_0})$.

3) L3 viva, se *t* aparece infinitas vezes em alguma seqüência de disparo de $L(\mathbf{M_0})$.

4) L4 viva ou simplesmente viva se *t* é L1 viva para cada marcação **M** em $R(\mathbf{M_0})$.

Uma rede $\mathbf{N=(S,T;F,K,W,M_0)}$ é Lk viva se cada transição $t \in \mathbf{T}$ é Lk viva, k=0, 1, 2, 3, 4. L4 é o nível de vivacidade mais forte e corresponde ao conceito de vivacidade expresso anteriormente.

São muitas as noções de equidade propostas na literatura sobre redes de Petri, porém uma coisa é clara, esta propriedade é uma medida da disparidade de ocorrência entre duas

transições tomadas duas a duas, numa rede.

Sendo assim temos dois conceitos básicos de equidade [Murata, 89]: equidade limitada e equidade incondicional. Duas transições estão numa relação de equidade limitada quando o número máximo de vezes que uma pode ocorrer sem que a outra ocorra é limitado. Neste caso uma rede é denominada de equidade limitada se cada par de transições na rede estão em uma relação de equidade limitada. Uma seqüência de disparo σ é dita incondicionalmente eqüitativa se a mesma é finita e se cada transição da rede aparece sempre que uma das outras ocorrer em σ. Uma rede é dita incondicionalmente eqüitativa se cada seqüência de disparo σ em $R(\mathbf{M_0})$ é incondicionalmente eqüitativa.

Uma rede $\mathbf{N=(S,T;F,K,W,M_0)}$ é dita limitada ou k-limitada se e somente se o número de marcas em cada lugar não supera um número finito k para qualquer marcação alcançável a partir de $\mathbf{M_0}$. Formalmente, $\forall s \in S: M(s) \leq k, k \in \mathbb{N}^+$.

Uma rede $\mathbf{N=(S,T;F,K,W,M_0)}$ é dita segura se e somente se o número de marcas em cada lugar é no máximo igual a 1, ou poderíamos dizer também se a mesma é 1-limitada. Formalmente, $\forall s \in \mathbf{S}: M(s) \leq 1$.

O comportamento dinâmico de muitos dos sistemas estudados pela engenharia podem ser descritos usando equações diferenciais. No caso das redes de Petri este comportamento dinâmico é regido pela equação de estado. Uma parte fundamental desta equação de estado é o que se denomina como matriz de incidência. Esta matriz de incidência representa a

estrutura da rede e pode ser descrita da forma seguinte:

$$A = [a_{ij}] \qquad \text{com } a_{ij} = a_{ij}^{+} - a_{ij}^{-}$$

onde:

$a_{ij}^{+} = W(i,j)$ é o peso do arco que leva do lugar i a transição j.

$a_{ij}^{-} = W(j,i)$ é o peso do arco que leva da transição j ao lugar i.

A matriz de incidência é então uma matriz de ordem *mxn* onde *m* é o número de lugares e *n* o número de transições.

A equação de estado numa rede é dada pela seguinte equação:

$$\mathbf{M_{k+1}} = \mathbf{M_k} + \mathbf{A}.\boldsymbol{\nu_k}$$

onde $\mathbf{M_k}$ e $\mathbf{M_{k+1}}$ são as marcações dos estados atual e sucessor respetivamente, e $\boldsymbol{\nu_k}$ é o vetor de habilitação.

Se consideramos que uma determinada marcação $\mathbf{M_f}$ é alcançável a partir de $\mathbf{M_0}$ através da seqüência de disparo $\{\boldsymbol{\nu_0}, \boldsymbol{\nu_2}, \ldots \boldsymbol{\nu_{f-1}}\}$ poderíamos expressar a equação de estado da seguinte forma:

$$\mathbf{M}_f = \mathbf{M}_0 + \mathbf{A}\sum_{k=0}^{f-1}\nu_k$$

a qual pode ser rescrita como:

$$\mathbf{A}.x = \Delta\mathbf{M}$$

onde $\Delta\mathbf{M} = \mathbf{M_f} - \mathbf{M_0}$ e $x = \sum \boldsymbol{v_k}$

O componente i-ésimo do vetor x de ordem nx1 denota a quantidade de vezes que a transição i deve ser disparada até alcançar a marcação $\mathbf{M_f}$.

Existem outras propriedades nas redes que podem ser calculadas a partir da matriz de incidência, a solução da equação $\mathbf{A}^T y=0$ nos permite conhecer o vetor de ponderação de marcas de modo que o produto deste vetor por qualquer vetor de marcação seja uma constante. Os vetores y que são solução desta equação são chamados de invariantes de lugar.

Seguindo o mesmo raciocínio a solução da equação $\mathbf{A}x=0$ teremos o número de vezes que um determinado conjunto de transições deve ser disparada para o sistema voltar a seu estado inicial. Os vetores x que são solução desta equação são chamados de invariantes de transição.

Existem situações numa rede que podem gerar confusão na hora de decidir uma determinada seqüência de disparos. Estas situações são chamadas de conflito e contato e são normalmente definidas sobre as redes condição/evento. A seguir definimos estas duas situações:

Seja uma rede **N=(B,E;F,C)**, dois eventos de **N** estão em conflito se e somente se eles possuem pré-condições válidas em comum. Formalmente:

$$\forall e_1, e_2 \in \mathbf{E}, e_1 \neq e_2, \forall c \in \mathbf{C} : {}^\bullet e_1, {}^\bullet e_2 \in c \wedge ({}^\bullet e_1 \cap {}^\bullet e_2) \neq \varnothing.$$

Seja uma rede **N=(B,E;F,C)**, um evento de **N** está em contato se e somente se suas pré-condições são válidas e ao menos uma de suas pós-condições também é válida. Formalmente:

$$\forall e \in \mathbf{E}, \forall c \in \mathbf{C}: ({}^\bullet e \subseteq c) \wedge (e^\bullet \cap c \neq \varnothing).$$

3.3.4 Redes estendidas.

A ausência do conceito de tempo nas redes de Petri, assim como de outros tipos de informação, além da falta de hierarquia que permita a verificação e análise em diferentes níveis de abstração, e a explosão combinatória de estados, são algumas das razões que deram origem às redes estendidas. Existem duas formas fundamentais para abordar estas limitações: introduzindo elementos sem alterar a teoria básica das redes ou modificando a teoria de redes. Nesta seção trataremos de forma genérica as propostas pertencentes ao primeiro grupo pois nossa proposta se encaixa neste grupo.

As redes estendidas são aquelas que introduzem novos elementos e abordagens sem alterar a teoria de redes, isto contribui para aumentar o poder de representação das redes sem perder o poder de análise das mesmas. Para tentar superar algumas limitações das redes de Petri na modelagem de sistemas muitas extensões tem sido propostas.

Entre as extensões mais conhecidas temos as seguintes:

Redes temporizadas: Neste caso as redes são incrementadas com o atributo de intervalos de tempo associados ao disparo das transições. Assim, o tempo, como intervalos discretos representa o fato de que os eventos e transições não são de fato instantâneos e devem reter a marca por um intervalo finito de tempo. Existem também as redes estocásticas, onde o tempo é função de modelos probabilísticos. Estas extensões são utilizadas para fazer a avaliação de desempenho de sistemas dinâmicos.

Extensão com elementos especiais: Outras redes foram estendidas com elementos especiais com comportamentos que simulam processos padrões realizados nas indústrias. Com estes novos elementos a modelagem de sistemas industriais ficou menos complicada, porém, na maioria dos casos, estas redes são úteis somente para um grupo seleto de aplicações. Exemplos deste tipo de redes são as redes MFG (Mark Flow Graph), E-MFG (Extended Mark Flow Graph).

Extensão com hierarquia: Como as extensões tendem a aumentar a precisão na especificação e modelagem de sistemas reais, existe uma tendência para que estes modelos se tornem cada vez maiores e mais complexos. Portanto seria interessante ter uma abordagem *top down* para a síntese de redes, especialmente aquelas que representam sistemas automatizados (que em geral envolvem um grau de complexidade maior). Para se obter uma síntese *top down* estruturada as redes precisam ter uma hierarquia. Para tanto, mesmo as redes coloridas estão incluindo o conceito de hierarquia [Jensen 90].

Extensão com orientação a objetos: Utiliza uma mistura destas abordagens de forma que o formalismo da teoria de redes não seja modificado. Difere portanto de outros modelos

onde a abordagem da orientação a objetos utilizada inclui marcas como objetos ficando assim mais perto das redes de alto nível.

3.3.5 Redes de alto nível.

A explosão combinatória de estados tem sido o grande motivador para que se procure a formalização de "dobramentos" nas redes. Deste modo, representações com um número menor de elementos, e com maior conteúdo semântico podem servir de base para a análise de sistemas, facilitando ao mesmo tempo o cálculo de algumas propriedades, tais como os invariantes e a distância síncrona. Historicamente, esta foi a motivação para os trabalhos originais de Genrich e Lautenbach [Genrich, 83], propondo a rede Predicado/Transição, e mais tarde os trabalhos de Kurt Jensen propondo a rede colorida [Jensen, 90].

No caso das redes Predicado/Transição, vários lugares na rede (condições) se fundem para dar origem a um lugar de ordem semântica superior cujas marcas são agora identificáveis. Similarmente, o mesmo processo é feito com as transições, que passam a admitir e modificar marcas que são funções e relações que agora fazem parte da definição da rede. Para se ter uma idéia do que isto pode representar em termos da complexidade da rede, tomemos o exemplo clássico dos três filósofos comendo (ou pensando) apresentado em [Reisig, 80] por uma rede com nove condições, seis eventos e 24 arcos. Esta mesma rede é representada por uma rede predicado transição (Pr/T) contendo três lugares, duas transições e apenas seis arcos (além das definições de domínio e de funções associadas às marcas).

No caso da rede colorida, o ponto de partida é a rede P/T (lugar/transição), onde as marcas

são identificadas através de tipos atribuídos a cada uma delas. As transições ganham mais poder de modelagem, já que podem receber marcas de um tipo e devolver marcas de tipo diferentes (no sentido da Teoria de Tipos), analogamente ao que acontece com o peso dos arcos antes e depois das transições nas redes P/T. O mesmo efeito de redução das redes é conseguido (desta vez o mesmo exemplo dos filósofos é reduzido para uma rede com apenas três elementos, dois lugares e uma transição, e mais as assinaturas e definições dos tipos e domínio de variáveis) [Jensen, 96].

Outras propostas de redes de alto nível têm surgido na literatura compondo a classe das redes orientadas a objetos [Lakos, 94], [Battiston, 90], [Esser, 97], [Sibertin-Blanc, 98].

Entretanto, o que se pode discutir em todas estas propostas é que apesar de todas as vantagens (inclusive os avanços na formalização de rede) persiste ainda o problema da síntese das redes. Com o desenvolvimento de novos sistemas cada vez existe um desafio maior na modelagem de sistemas integrados, flexíveis, automatizados, etc., e o problema é obter um modelo formalizado em redes de Petri, para se proceder ao processo de análise e cálculo de propriedades. Por outro lado, a maioria das propostas de extensão abdicam do formalismo já existente em função da inclusão de elementos de interpretação direta, enquanto que as redes de alto nível procuram manter o formalismo, um "dobramento" atraente, mas que afastam o modelo da sua interpretação (como acontecem na maioria dos esquemas e nas linguagens formais de especificação e *design*) [Silva, 96][Silva98].

4 DEFINIÇÃO DA REDE GHENESYS E OS SEUS ELEMENTOS.

Neste capítulo apresenta-se a definição formal da rede GHENeSys (General Hierarchical Enhanced Net System). Fazendo um esquema hierárquico para representar a evolução das redes de Petri (das redes elementares até as redes de alto nível) e algumas das suas extensões mais citadas na literatura podemos colocar as redes GHENeSys entre as redes condição/evento e lugar/transição.

Na verdade a rede GHENeSys é derivada do MFG [Hasegawa, 87], e da proposta de síntese *top down* das redes introduzida no PFS/MFG [Miyagi, 88], da proposta de transformação do método PFS/MFG em uma rede [Silva, 92][Silva, 96], e finalmente na formulação orientada a objetos apresentada neste trabalho. A proposta apresentada aqui é de uma rede estendida, devido a inclusão de características tais como multiplicidade de marcas, hierarquia distribuída nos elementos ativos e passivos e fluxo de informação através de relações especiais chamadas *gates* que transmitem informação de eventos externos á rede. Estas relações especiais são representadas através de um elemento especial chamado *pseudobox* que permite a representação de eventos não controláveis [Lin&Wonham, 90], ou

mais genericamente, condições pertinentes ao contexto em que o modelo se encontra (partindo do pressuposto de que não existe sistema realmente isolado). Estes elementos podem também ser usados para representar transferência de informação entre partes integrantes da mesma rede.

A representação de todos os elementos básicos da rede é feita na forma de objetos, embora na proposta apresentada aqui ainda não se represente a marca desta forma, e portanto não se possa reclamar o título de rede estendida para o GHENeSys.

4.1 Definição da rede GHENeSys.

Na definição da rede serão utilizadas as notações freqüentemente usadas na teoria de redes.

DEFINIÇÃO 1. Uma rede GHENeSys é uma quíntupla **N=(L, A, F, K, M)** onde:

i) $\mathbf{L} \cap \mathbf{A} = \varnothing$

ii) $\mathbf{L} \cup \mathbf{A} \neq \varnothing$

iii) $\mathbf{F} \subseteq (\mathbf{L}\mathrm{x}\mathbf{A}) \cup (\mathbf{A}\mathrm{x}\mathbf{L})$

iv) $\mathbf{B} \cup \mathbf{P} = \mathbf{L}$

v) $\mathbf{K}: \mathbf{L} \rightarrow \mathbb{N}^{+}$ e para todo $p \in \mathbf{P}$, $\mathrm{K}(p) = 1$.

vi) $\mathbf{M}: \mathbf{L} \rightarrow \mathbb{N}^{+}$ sendo que: $\mathbf{M}(l) \leq \mathbf{K}(l)$ para todo $l \in \mathbf{L}$

Os elementos do conjunto **L** são chamados de lugares e segundo o item *iv* é composto pela união dos conjuntos **B** e **P** (*boxes* e *pseudoboxes* respectivamente). Os elementos do conjunto **A** são chamados de atividades. **F** é a relação de fluxo e os seus elementos são chamados de arcos. **K** é a função de capacidade e indica a capacidade máxima permitida em cada lugar, sendo que os *pseudoboxes* tem capacidade unitária. **M** é a marcação inicial da rede respeitando as capacidades de cada lugar.

Os elementos do conjunto **L** têm funções diferentes na equação de estado da rede: os pseudoboxes **P** representam lugares com marcação persistente[2] e isto é considerado na equação de estado que trataremos na seção seguinte.

Levando em consideração o item *iv* podemos rescrever o item *iii* da seguinte forma:

$$\mathbf{F} \subseteq (\mathbf{BxA}) \cup (\mathbf{AxB}) \cup (\mathbf{PxA}) \cup (\mathbf{AxP})$$

Os dois primeiros termos representam os arcos que conectam os boxes com as atividades, e têm a mesma interpretação que nas redes de Petri elementares. Estes arcos permitem a passagem de uma marca por vez, o que faz com que a matriz de incidência para esta rede seja unitária. O terceiro termo representa a conexão dos pseudoboxes com as atividades. Estas relações não podem ser interpretadas da mesma forma, isto é, não pressupõe o fluxo

---

[2] Marcação que não é modificada durante a execução do disparo.

de marcas, mas somente a informação sobre uma marcação persistente em um dos pseudoboxes. Estas conexões podem ser de dois tipos: habilitadoras (permitem o disparo quando estão marcadas) ou inibidoras (permitem o disparo quando não estão marcadas). O último termo carece de sentido prático na nossa rede assim pode ser removido ficando a relação de fluxo definida da seguinte forma:

$$\mathbf{F} \subseteq (\mathbf{B}\mathrm{x}\mathbf{A}) \cup (\mathbf{A}\mathrm{x}\mathbf{B}) \cup (\mathbf{P}\mathrm{x}\mathbf{A})$$

Sendo

$$\mathbf{F} = \begin{cases} -1 & \text{para } (\mathbf{B}\mathrm{x}\mathbf{A}) \\ 1 & \text{para } (\mathbf{A}\mathrm{x}\mathbf{B}) \\ 1 & \text{para } (\mathbf{P}\mathrm{x}\mathbf{A}) \text{ sendo habilitadora} \\ -1 & \text{para } (\mathbf{P}\mathrm{x}\mathbf{A}) \text{ sendo inibidora} \end{cases}$$

Os elementos do subconjunto **P** representam em princípio a informação externa à rede em questão mas, nada impede que este elemento represente um elemento do conjunto **B** de outra rede, quando a rede é dividida em módulos hierarquizados. A utilidade óbvia deste elemento é modelar a transmissão de informação entre diferentes partes de um mesmo sistema assim como a influência da informação externa.

A seguir apresentam-se algumas definições básicas da nossa rede partindo das já existentes sobre redes de Petri [Thiagarajan, 87], [Reisig, 82].

DEFINIÇÃO 2. Dada uma rede **N=(L, A, F, K, M)** com $\mathbf{X} = \mathbf{L} \cup \mathbf{A}$ e $x \in \mathbf{X}$, definimos:

$$^{\bullet}x = \{ y \in \mathbf{X} \mid (x, y) \in \mathbf{F} \}$$

e $^{\bullet}x$ é o conjunto dos pré-elementos de x.

$$x^{\bullet} = \{ y \in \mathbf{X} \mid (x, y) \in \mathbf{F} \}$$

e $x^{\bullet}$ é o conjunto dos pós-elementos de x.

DEFINIÇÃO 3. Dada uma rede **N=(L, A, F, K, M)**:

i) A aplicação **M**: $\mathbf{L} \rightarrow \mathbb{N}^{+}$ é chamada uma marcação em **N** se e somente se $\mathbf{M}(l) \leq \mathbf{K}(l)$ para todo $l \in \mathbf{L}$. Chamaremos *case*[3] **C** ao conjunto de elementos $l \in \mathbf{L}$ tal que l esteja presente na marcação corrente **M**, Formalmente:

$l \in \mathbf{C}$ se e somente se $l \in \mathbf{L} | \mathrm{M}(l) \geq 1$

Chama-se *case set* $\Im$ ao conjunto de todos os possíveis *cases* para uma estrutura de rede.

ii) Seja **M** uma marcação em **N**, podemos dizer que uma atividade $a \in \mathbf{A}$ está habilitada se e somente se:

- $\forall b \in {}^{\bullet}a \,.\, b \in \mathbf{B} \wedge b \in \mathbf{M}$, isto é, todos os boxes pertencentes aos pre-elementos de a têm pelo menos uma marca;
- $\forall b \in a^{\bullet}.\ \mathbf{M}(b) \leq \mathbf{K}(b) - 1$, isto é, todos os boxes pertencentes aos pós-elementos de a têm a capacidade de absorver pelo menos uma marca;

---

[3] Apesar de ter o mesmo sinificado que nas redes condição/evento a definição não é a mesma devido a que a rede GHENeSys admite marcação múltipla.

- $\forall p \in {}^{\bullet}a$, $p \in \mathbf{M}$ se p é um pseudobox habilitador, $p \notin \mathbf{M}$ se p é um pseudobox inibidor;

iii) Uma marcação **M** que habilita a leva a uma marcação **M'**, onde

M' = $[\mathbf{M} \backslash ({}^{\bullet}a \cap \mathbf{B})] \cup (a^{\bullet} \cap \mathbf{B})$ e pode ser representado como $\mathbf{M}[a>\mathbf{M'}$. Isto indica que a marcação persistente dos pseudoboxes não é alterada com o disparo;

iv) Definimos R(**M**) como sendo o conjunto das marcações de **N** alcançáveis a partir de **M** tal que:

1) $\mathbf{M} \in$ R(**M**) e

2) se $\mathbf{M_1} \in$ R(**M**) e para algum $a \in \mathbf{A_N}$ $\mathbf{M_1}[a>\mathbf{M_2}$ então $\mathbf{M_2} \in$ R(**M**).

Na figura 4.1 aparece a representação gráfica dos elementos da rede GHENeSys

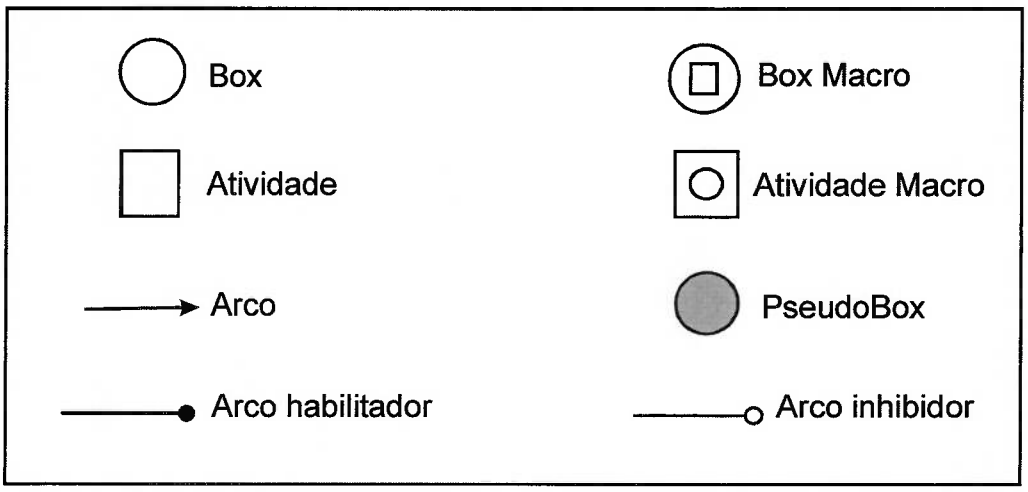


Fig 4.1 Representação gráfica dos elementos que compõem a rede GHENeSys.

Na figura 4.2 é mostrado um exemplo de rede GHENeSys. Neste exemplo assume-se que para os boxes onde não foi especificado a capacidade a mesma e igual a 1.

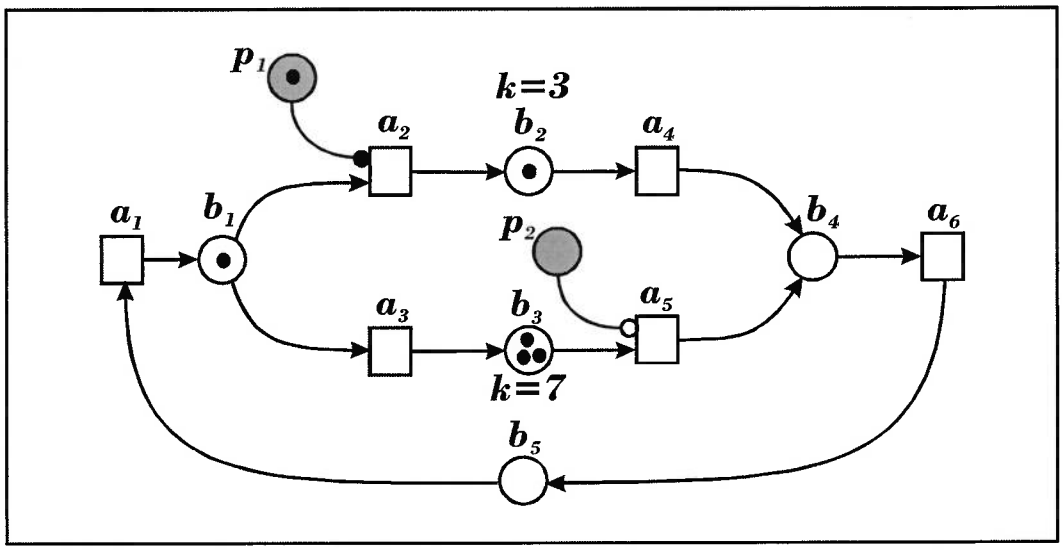


Fig 4.2 Um exemplo de rede GHENeSys

4.1.1 Matriz de incidência.

A matriz de incidência de uma rede depende da relação de fluxo, portanto para qualquer lugar l e qualquer atividade a poderão existir quatro tipos de relações:

- $(l,a) \notin \mathbf{F}$ e $(a,l) \notin \mathbf{F}$. Significa que l não tem nenhuma influência na habilitação de a e por sua parte a não muda o número de marcas em l.

- $(l,a) \in \mathbf{F}$ e $(a,l) \notin \mathbf{F}$. Neste caso a pode estar habilitada se existe ao menos uma marca em l e a ocorrência de a diminui o número de marcas de l em 1.

- $(l,a) \notin \mathbf{F}$ e $(a,l) \in \mathbf{F}$. Neste caso, a ocorrência de a é possível só se o lugar l aceita mais uma marca, a ocorrência de a aumenta o número de marcas de l em 1.

- $(l,a)\in\mathbf{F}$ e $(a,l)\in\mathbf{F}$. Neste caso a pode estar habilitada se existe ao menos uma marca em l e a ocorrência de a não muda o número de marcas em l. Na rede GHENeSys este caso não é permitido, a rede GHENeSys é por definição uma rede pura pois não admite a presença de laços.

Sendo assim, a matriz de incidência C:(LxA)→{-1,0,1} de **N** seria definida por:

$$C(l,a) = \begin{cases} 0 \text{ se } (l,a)\notin\mathbf{F} \text{ e } (a,l)\notin\mathbf{F} \\ -1 \text{ se } (l,a)\in\mathbf{F} \text{ e } (a,l)\notin\mathbf{F} \\ 1 \text{ se } (l,a)\notin\mathbf{F} \text{ e } (a,l)\in\mathbf{F} \end{cases}$$

Seguindo esta definição, a matriz de incidência da figura 4.2 ficaria da seguinte forma:

$$C = \begin{array}{c} \begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{matrix} \\ \begin{bmatrix} 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ p_1 \\ p_2 \end{matrix}$$

4.1.2 Equação de estado.

Na rede GHENeSys, analogamente às redes de Petri, um novo estado depende do estado em que a rede está e das atividades (transições nas redes de Petri elementares) habilitadas nesse momento. Podemos dizer que o estado da rede (marcação de um conjunto de lugares

da rede) depende linearmente do estado inicial (marcação inicial) e da seqüência em que foram sendo executadas as atividades uma vez habilitadas.

Como na matriz de incidência aparecem os pseudoboxes e estes são elementos com marcação persistente é necessário fazer uma modificação na equação de estado das redes de Petri elementares. Esta modificação prevê o uso de uma matriz **D** que é uma matriz de ordem (mxm) sendo m o número de lugares da rede. Esta matriz é uma matriz diagonal cuja diagonal principal corresponde ao vetor coluna *d* definido como : d_{1i}=1 se *i* é um *box*, e d_{1i}=0 se *i* é um *pseudobox*.

A seguir apresenta-se o vetor *d* e a matriz **D** correspondente ao exemplo da figura 4.2.

$$d = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0]$$

$$D = \left[\begin{array}{ccccc:cc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \begin{array}{l} \\ \\ \\ \\ \\ \text{B} \\ \\ \text{P} \end{array}$$

A equação de estado modificada é a seguinte:

$$\mathbf{M}_{k+1} = \mathbf{M}_k + \mathbf{DC}\nu_k$$

onde ν_k é o vetor de habilitação no estado $\mathbf{M}_k$.

A seguir mostraremos um algoritmo para calcular o vetor de habilitação.

Para determinar o vetor de habilitação seguiremos os seguintes passos:

Primeiramente multiplicamos $\mathbf{M}_k$ (vetor de marcação) por uma matriz linha composta de 1's, de ordem (1xn) onde n é o número de atividades da rede. O resultado desta multiplicação e uma matriz de ordem (mxn) (a mesma ordem que a matriz de incidência **C**) que denominaremos como $\mathbf{M}_a$.

Em seguida, somamos a matriz $\mathbf{M}_a$ com a matriz **C** resultando na matriz que denominamos como $\mathbf{M}_i$.

Note que cada coluna, as quais denominaremos de $\mathbf{v}_i$, da matriz $\mathbf{M}_i$ representa a marcação resultante do disparo da atividade a_i, portanto para determinar se esta atividade estaria habilitada devemos comparar esta possível marcação resultante com uma marcação válida segundo a definição 3ii). Para isto precisaremos da seguinte definição.

DEFINIÇÃO 4. Um vetor **A** é maior ou igual a um vetor **B** (**A≥B**) se e somente se:

i) **A** e **B** são da mesma ordem.

ii) O elemento *i* de **A** é maior ou igual ao elemento *i* de **B**:

$a_i \geq b_i$, onde $1 \leq i \leq n$, onde *n* é o número de elementos de **A** e **B**.

PROPOSIÇÃO 1. Uma atividade *a* está habilitada se o somente se o seu vetor coluna correspondente $\mathbf{v}_i$ satisfaz a desigualdade $\mathbf{v}_0 \leq \mathbf{v}_i \leq \mathbf{K}^T$ onde $\mathbf{v}_0$ é um vetor da mesma ordem de $\mathbf{v}_i$ com todos seus elementos iguais a 0 e **K** é o vetor das capacidades.

A matriz $\mathbf{M}_i$ é dada pela expressão

$$\mathbf{M}_i = \mathbf{M}_k \mathbf{I} + \mathbf{C}$$

No exemplo mostrado acima, o número de lugares é 7 (m=7, incluindo os pseudoboxes) e de atividades é 6 (n=6), portanto a matriz denominada como **I** seria a seguinte:

$$\mathbf{I} = [\,1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1\,]$$

A matriz $\mathbf{M}_i$ resultante desta operação seria:

$$M_i = \begin{bmatrix} 2 & 0 & 0 & 1 & 1 & 1 \\ 1 & 2 & 1 & 0 & 1 & 1 \\ 3 & 3 & 4 & 3 & 2 & 3 \\ 0 & 0 & 0 & 1 & 1 & -1 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Se cada um dos vetores coluna é comparado segundo a proposição 1 pode-se determinar claramente que o vetor de habilitação é o seguinte:

$$\nu_k = [\,0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0\,]^T$$

O fato do vetor de habilitação acima indicar que as atividades a_2, a_3, a_4, a_5 estão habilitadas

não significa necessariamente que estas possam acontecer simultaneamente. Se existe alguma situação de conflito ou contato na rede que estamos analisando gostaríamos de saber quais são as atividades envolvidas nesta situação.

A seguir apresentaremos um algoritmo para detectar situações de conflito e contato. Este algoritmo é necessário se pensarmos num sistema que dispara o maior número possível de atividades, uma vez que estas estejam habilitadas. Os casos de conflito poderiam ser resolvidos utilizando um algoritmo de inteligência artificial ou com a intervenção direta do usuário.

Para detectar situações de conflito ou contato primeiro devemos calcular o vetor de habilitação utilizando o algoritmo anterior. Uma vez calculado o vetor de habilitação passamos ao cálculo a matriz $\mathbf{M}_c$ usando a seguinte equação.

$$\mathbf{M}_c = \mathbf{M} + \mathbf{C}v_k$$

A matriz resultante é uma matriz coluna de ordem (mx1).

PROPOSIÇÃO 2. O vetor de conflito que denominamos como $\mathbf{v}_c$ é determinado segundo a seguinte regra:

$$[v_c]_{j1} = \begin{cases} -1 & \text{se } [\mathbf{M}_c]_{j1} < 0 \\ 1 & \text{se } [\mathbf{M}_c]_{j1} > K(j) \\ 0 & \text{resto dos casos.} \end{cases}$$

Portanto, assim que o vetor $\mathbf{v}_c$ é obtido é possível determinar quais dos vetores habilitados estão em conflito, determinando a matriz $\mathbf{C}_c$ segundo a seguinte equação:

$$\mathbf{C}_c = \mathbf{C} + (\nu_k \, \mathbf{v}_c)^T$$

Esta matriz $\mathbf{C}_c$ contém a informação necessária para detectar os conflitos usando a seguinte definição.

DEFINIÇÃO 5. Dada a matriz de conflito $\mathbf{C}_c$

i) Para um lugar ℓ_i, as atividades a_j tais que $[\mathbf{C}_c]_{i,j} = -2$ estão em conflito devido ao compartilhamento do lugar ℓ_i, isto é, $\ell_i \in \cap_j {}^\bullet a_j$ (conflito de pré-elemento);

ii) Para um lugar ℓ_i, as atividades a_j tais que $[\mathbf{C}_c]_{i,j} = 2$ estão em conflito por compartilhamento das pós-elemento, isto é, $\ell_i \in \cap_j a_j^\bullet$;

iii) Nos demais casos, $[\mathbf{C}_c]_{i,j} = [\nu]_{lj}[\mathbf{v}_c]_{li}$.

As situações de contato não foram levadas em conta porque sempre é possível calcular a matriz completa, incluindo os elementos complementares [Reisig 82] onde o contato seria eliminado. No caso da rede GHENeSys, eliminando-se os pseudoboxes, os demais elementos se comportam como condições com multiplicidade de marcas o que se encaixa na definição de dual dada por [Murata, 89], para elementos da rede lugar/transição.

No exemplo da figura 4.1 a matriz $\mathbf{C}_c$ é a seguinte:

$$C_c = \begin{bmatrix} 1 & -2 & -2 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 2 & 2 & -1 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Desta forma fica claro que as atividades a_2 e a_3 estão em conflito devido ao lugar b_1 e, a_4 e a_5 estão em conflito devido ao lugar b_4.

Os programas desenvolvidos no Matlab para realizar estes cálculos são listados no Apêndice I.

4.1.3 Algumas definições básicas para modelar comportamento na rede GHENeSys.

Antes de definir as propriedades da rede GHENeSys, apresentam-se algumas definições úteis que serão utilizadas neste capítulo.

DEFINIÇÃO 6. Seja **N = (L, A, F, K, M_0)** uma rede

i) Uma marcação $\mathbf{M}_n$ é dita alcançável a partir de $\mathbf{M}_0$ se existe uma seqüência de disparo de atividades que levam de $\mathbf{M}_0$ a $\mathbf{M}_n$:

$$\mathbf{M}_0 \xrightarrow{a_1} \mathbf{M}_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} \mathbf{M}_n$$

ii) A seqüência de disparo é denotada por $\sigma = a_1\, a_2 \ldots a_n$

Neste caso dizemos que $\mathbf{M}_n$ é alcançável à partir de $\mathbf{M}_0$ e escreve-se $\mathbf{M}_0[\sigma>\mathbf{M}_n$

iii) O conjunto de todas as possíveis seqüências de disparo de **N** a partir de $\mathbf{M}_0$ é denominada como $L(\mathbf{M}_0)$.

DEFINIÇÃO 7. Uma rede $\mathbf{N} = (\mathbf{L}, \mathbf{A}, \mathbf{F}, \mathbf{K}, \mathbf{M}_0)$ é denominada cíclica se e somente se:

$$\forall \mathbf{M}_1, \mathbf{M}_2 \in R(\mathbf{M}_0), \exists \sigma \in L(\mathbf{M}_0) \mid \mathbf{M}_1[\sigma^*>\mathbf{M}_2.$$

DEFINIÇÃO 8. Uma rede $\mathbf{N} = (\mathbf{L}, \mathbf{A}, \mathbf{F}, \mathbf{K}, \mathbf{M}_0)$ é denominada finita se e somente se:

i) $\mathbf{L} \cup \mathbf{A} \neq \infty$, e

ii) **N** não é cíclica.

Logo das definições e notações anteriores, estamos prontos para apresentar as definições das propriedades da rede.

A vivacidade de uma rede depende da marcação inicial. Uma marcação $\mathbf{M}_0$ é dita viva na rede **N** se e somente se, independentemente da marcação alcançada partindo de $\mathbf{M}_0$ é possível disparar qualquer atividade em **N** pelo menos uma vez. Isto aparece formalmente expresso na seguinte definição:

DEFINIÇÃO 9. Seja $\mathbf{N} = (\mathbf{L}, \mathbf{A}, \mathbf{F}, \mathbf{K}, \mathbf{M}_0)$ uma rede

i) $a \in \mathbf{A}$ é viva se e somente se $\forall \mathbf{M} \in R(\mathbf{M}_0)\ \exists \mathbf{M}_1 \in R(\mathbf{M}_0) \mid \mathbf{M}[a>\mathbf{M}_1$

iii) **N** é viva se e somente se $\forall a \in \mathbf{A}$. a é viva.

DEFINIÇÃO 10. Seja uma rede $\mathbf{N}=(\mathbf{L},\mathbf{A},\mathbf{F},\mathbf{K},\mathbf{M}_0)$ temos que:

i) Uma atividade $a \in \mathbf{A}$ é dita:

1. morta (L0-viva), se a não aparece em nenhuma seqüência de disparo de $L(\mathbf{M_0})$.
2. Potencialmente disparável (L1-viva), se a aparece ao menos uma vez em alguma seqüência de disparo de $L(\mathbf{M}_0)$.
3. L2 viva se dado um número $k \in \mathbb{N}^+$, a aparece ao menos k vezes em alguma seqüência de disparo de $L(\mathbf{M}_0)$.
4. L3 viva se a aparece infinitas vezes em alguma seqüência de disparo de $L(\mathbf{M}_0)$.
5. L4 viva ou simplesmente viva se a é L1 viva para cada marcação M em $R(\mathbf{M}_0)$.

ii) A rede **N** é Lk viva se cada atividade $a \in \mathbf{A}$ é Lk viva, k=0, 1, 2, 3, 4. L4 é o nível de vivacidade mais forte e corresponde ao conceito de vivacidade expresso na definição 9.

Baseado na definição anterior dos distintos graus de vivacidade podemos facilmente chegar à seguinte conclusão:

L4 viva $\Rightarrow$ L3 viva $\Rightarrow$ L2 viva $\Rightarrow$ L1 viva.

Atendo-se estritamente à definição destes diferentes níveis de vivacidade, dizemos que uma determinada rede **N** com marcação inicial $\mathbf{M}_0$ é Lk-viva se a mesma não é L(k+1) viva, k=0, 1, 2, 3.

Partindo da afirmação anterior podemos chegar à seguinte definição:

DEFINIÇÃO 11. Uma rede **N** é Lk-viva se e somente se:

i) **N** não é L(k+1) viva, k=0, 1, 2, 3.

ii) **N** é também Lj-viva, $\forall j.\ 0<j\leq k$, k=0, 1, 2, 3.

A partir da definição de equidade apresentada em [Murata 89] podemos definir a propriedade equivalente para a rede GHENeSys.

DEFINIÇÃO 12 Seja uma rede $\mathbf{N=(L,A,F,K,M_0)}$ temos que:

i) Duas atividades a_1 e a_2 estão numa relação de equidade limitada, se e somente se o número máximo de vezes que uma pode ocorrer sem que a outra ocorra é limitado.

ii) A rede **N** é denominada de equidade limitada, se e somente se, cada par de atividades da rede estão em uma relação de equidade limitada.

DEFINIÇÃO 13 Seja uma rede $\mathbf{N=(L,A,F,K,M_0)}$ temos que:

i) Uma seqüência de disparo σ é dita incondicionalmente eqüitativa, se e somente se a mesma é finita ou se cada atividade $a_i \in \mathbf{A}$ aparece sempre que uma das outras atividades ocorrer em σ.

ii) A rede **N** é dita incondicionalmente eqüitativa, se e somente se cada seqüência de disparo σ em $R(\mathbf{M_0})$ é incondicionalmente eqüitativa.

Outra propriedade importante é a de rede segura. Para definir segurança, primeiramente devemos definir quando uma rede é limitada ou k-limitada.

DEFINIÇÃO 14 Seja uma rede $\mathbf{N}=(\mathbf{L},\mathbf{A},\mathbf{F},\mathbf{K},\mathbf{M}_0)$ e $k \in \mathbb{N}^+$, temos que:

i) $\ell \in \mathbf{L}$ é k-limitado se e somente se $\forall \mathbf{M} \in L(\mathbf{M}_0) : M(\ell) \leq k$.

ii) $\ell \in \mathbf{L}$ é limitado se e somente se $\exists k$ tal que ℓ é k-limitado.

iii) **N** é k-limitada se e somente se $\forall \ell \in \mathbf{L}$. ℓ é k-limitado.

Agora podemos definir segurança:

DEFINIÇÃO 15 Seja uma rede $\mathbf{N}=(\mathbf{L},\mathbf{A},\mathbf{F},\mathbf{K},\mathbf{M}_0)$, temos que:

i) $\ell \in \mathbf{L}$ é seguro se e somente se ℓ é 1-limitado.

ii) **N** é segura se e somente se $\forall \ell \in \mathbf{L}$, ℓ é seguro.

Existem várias outras propriedades que podem ser consideradas na hora de fazer uma análise do comportamento das redes e dos sistemas que estes representam. Entre estas outras propriedades podemos colocar a distância síncrona, e os invariantes de lugar e atividades. Estas duas últimas, são consideradas propriedades estruturais já que não dependem da marcação inicial da rede.

Estas propriedades serão tratadas no próximo item.

4.2 Elementos de extensões da rede GHENeSys.

Ao colocar a definição da rede GHENeSys alguns de seus elementos estendidos já foram introduzidos, no caso, os pseudoboxes. Os *pseudoboxes* foram introduzidos, primeiramente como um elemento de singular importância na hora de modelar transferência de informação entre diferentes partes de um sistema, permitindo assim a modularização e conseqüente estruturação do processo de síntese da rede. Uma outra razão é dotar as redes da capacidade de representar a interação com o ambiente que envolve os sistemas, já que, como salientamos no capítulo anterior, não existem de fato sistemas isolados na natureza.

Estes elementos são ainda utilizados para representar informações de eventos observáveis, previsíveis, porém não controláveis [Lin&Wonhan, 90] [Ramos, 98].

Outros conceitos úteis na estruturação da síntese de uma rede GHENeSys serão tratadas a seguir.

4.2.1 Hierarquia.

Existem na literatura diferentes métodos de refinamento associado a redes de Petri [Fehling, 93], [Valette, 79], [Vogler, 87]. Estes métodos são imprescindíveis para conseguir representar sistemas complexos e são também a base para procedimentos estruturados de síntese de sistemas.

No projeto de sistemas, utilizando uma abordagem *top-down*, o modelo é construído partindo de uma abstração da rede que representa o sistema (composta por módulos principais e seus relacionamentos) e, à partir desta, os módulos são refinados até atingir um nível de detalhe adequado para a análise e implementação.

Para diminuir a influência dos possíveis erros de *design* (principalmente na fase inicial de modelagem) seria razoável poder contar com a possibilidade de analisar o modelo nos diferentes níveis de refinamento. Desta forma os erros seriam rapidamente detectados e corrigidos antes de prosseguir com os refinamentos.

Neste sentido, aproveitando os resultados obtidos em [Valette 79] e [Suzuki 83], e adequando-os à definição da nossa rede, definimos elementos "macros" que são a base da hierarquia na rede GHENeSys.

Os elementos chamados "macros" são elementos que representam uma subrede. Como uma rede é originada de um grafo bipartido, os elementos "macros" podem ser classificados em dois tipos: atividades (os elementos ativos) ou lugares (os elementos passivos). As macro atividades começam e terminam com atividades (eventos) e os macro lugares começam e terminam com lugares. Assim, nossa rede pode ser construída mesclando elementos simples e macros nos diversos níveis de abstração. Esta facilidade de agregação está de acordo com a evolução do processo de modelagem e *design*, onde elementos definidos abstratamente convivem com outros definidos em detalhe, eventualmente reutilizados de outros projetos.

Os elementos "macros" devem atender a definição de elemento próprio baseado na definição de programa próprio apresentada em [Linger, 79]. A seguir apresentamos a definição de elemento macro.

DEFINIÇÃO 16. Um elemento macro D que pertence a uma rede $\mathbf{N}=(\mathbf{L},\mathbf{A},\mathbf{F},\mathbf{K},\mathbf{M}_0)$ onde $\mathbf{B} \cup \mathbf{P} = \mathbf{L}$, é representado por uma subrede $\mathbf{Nr}=(\mathbf{Lr},\mathbf{Ar},\mathbf{Fr},\mathbf{Kr},\mathbf{Mr}_0)$ onde $\mathbf{Br} \cup \mathbf{Pr} = \mathbf{Lr}$ com marcação final $\mathbf{Mr}_n$ onde:

i) **Nr** tem dois (únicos) elementos especiais: i e o, que cumprem que ${}^\bullet i = \varnothing$, $o^\bullet = \varnothing$.

ii) Se $D \in \mathbf{B}$ então $i \in \mathbf{Br}$ e $o \in \mathbf{Br}$, e se $D \in \mathbf{A}$ então $i \in \mathbf{Ar}$ e $o \in \mathbf{Ar}$.

iii) $\mathrm{dom}(\mathbf{Fr}) \cup \mathrm{cod}(\mathbf{Fr}) = \mathbf{Lr} \cup \mathbf{Ar}$

iv) Se $D \in \mathbf{B}$ então $\mathbf{Mr}_0(i) > 0$ e $\mathbf{Mr}_n(o) > 0$.

v) Se $D \in \mathbf{A}$ então $\exists \mathbf{M} \in [\mathbf{M}_0\rangle \mid \mathbf{M}[i\rangle \mathbf{Mr}_0$, e $\exists \mathbf{M}' \in [\mathbf{Mr}_0\rangle \mid \mathbf{M}'[o\rangle \mathbf{Mr}_n$.

A definição de elemento macro mostra através dos itens ii) e iv) que os elementos do conjunto **P** (*Pseudoboxes*) não podem ser refinados nem podem aparecer como elementos de entrada ou saída nos elementos macro.

Esta definição de elemento macro contém algumas caraterísticas "desejáveis" (estruturadas) tais como: possuir somente uma entrada e uma saída (i), poder representar elementos tanto estáticos quanto dinâmicos (ii), e não possuir elementos não conectados (iii). Estas caraterísticas ainda não são suficientes para que estes elementos preservem as propriedades

estruturais e comportamentais. Para tanto é necessário introduzir a seguinte definição.

DEFINIÇÃO 17. Um elemento macro D é dito bem construído se e somente se:

i) D é um elemento macro segundo a Definição 16.

ii) A rede Nr que representa D é viva.

Para simplificar a notação, qualquer citação a elemento macro neste trabalho deve ser entendida como elemento macro bem construído uma vez que estamos interessados na construção de sistemas com representação estruturada, passíveis de um processo de síntese *top down*, mesclando elementos abstratos e detalhados, e finalmente, permitindo a reutilização de outros elementos macro.

Em seguida descreveremos como seria, do ponto de vista da representação algébrica, um processo de refinamento de um elemento macro da rede.

Com o refinamento de um elemento aparecem novos elementos no seu lugar que não possuem relação alguma com os elementos da rede antes do refinamento, a não ser pela aparição de dois elementos do mesmo tipo do elemento macro que assumem as relações de entrada e saída do elemento macro. O resto dos elementos do refinamento só se relacionam entre si e os elementos de entrada e saída.

Nas figuras 4.3 e 4.4 podemos ver um exemplo de rede com elemento macro e o resultado do seu refinamento com as suas respetivas matrizes de incidência.

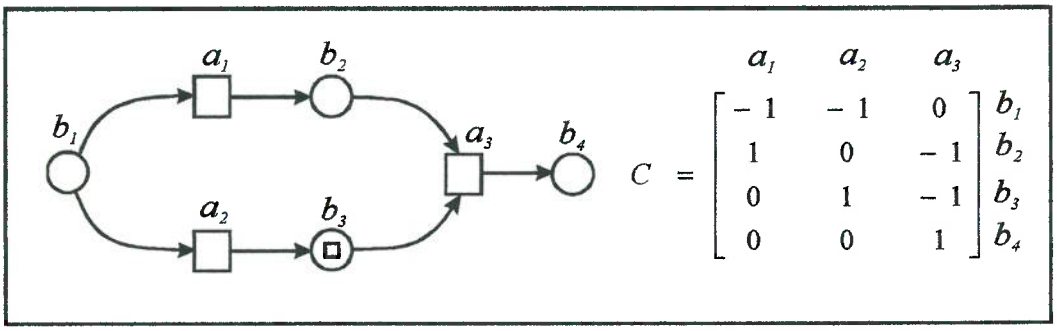


Fig 4.3 Uma rede com elemento "macro".

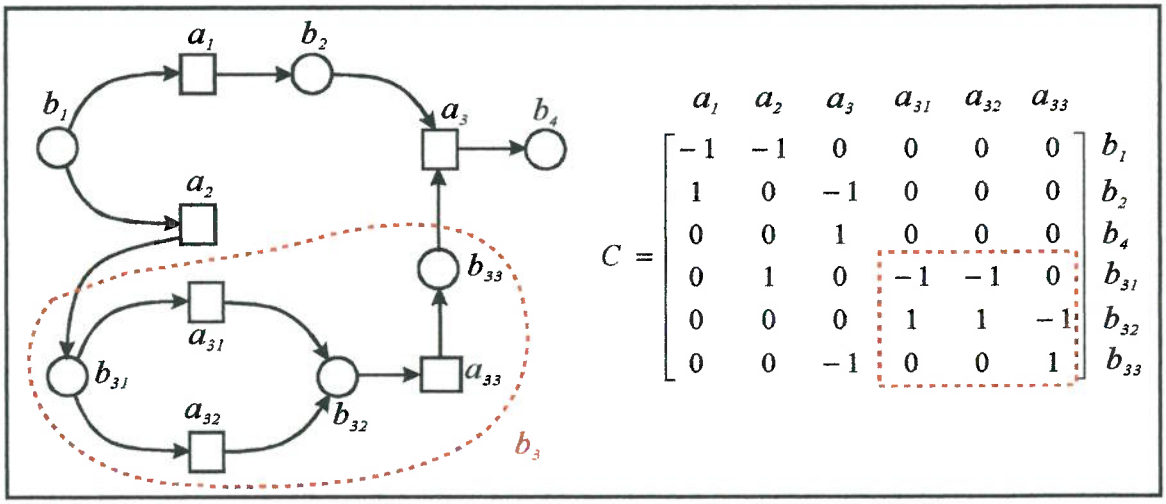


Fig. 4.4 Resultado do refinamento do elemento b_3.

Como podemos ver, na matriz de incidência, apesar do aumento de elementos da rede e portanto do ordem da matriz, a mesma não teve modificação nas filas correspondentes aos elementos da rede sem refinamento, salvo os elementos b_{31} e b_{33} que absorvem as relações do elemento b_3 na rede da figura 4.3. (Note-se que na matriz acima as linhas foram trocadas de modo que o quadro inferior direito representa a subrede que substitui o elemento b_3).

Um comportamento similar acontece quando refinamos elementos "macros" que são atividades. Estes elementos começam e terminam com atividades e modificam a matriz de

incidência de forma similar ao caso dos lugares, exceto que os elementos que absorvem a relação com o resto da rede são atividades em vez de lugares.

4.2.2 Orientação a objetos.

Além da introdução de estruturação no processo de modelagem, a proposta apresentada neste trabalho também pretende sintetizar as diversas possibilidades de extensão na rede, partindo das classes genéricas lugar e atividade.

Como em outras propostas os elementos da rede são objetos, e as marcas são objetos passivos (não contém métodos). A rede representa a estrutura de controle do sistema, enquanto as marcas modelam a estrutura de dados do sistema só que estes dados não aparecem nas regras de disparo das transições e portanto não alteram radicalmente o formalismo da rede, como é o caso das redes Pr/T e coloridas.

Os elementos da rede, que pertencem a uma determinada classe, podem ter subredes conectadas a eles. No caso destes elementos representarem estruturas amplamente conhecidas e que são freqüentemente usadas nos modelos, as subredes que eles representam podem ser substituídos por métodos da classe. Um exemplo deste caso é a aplicação de uma disciplina para gerenciar a chegada de elementos em um *box*, introduzindo o FIFO (First in first out), usado em várias extensões que distinguem as marcas por *tags*, para modelar por exemplo, fluxo de mensagens em redes e na Internet.

Estes métodos representam determinados comportamentos muito bem identificados e que não alteram as propriedades da rede como por exemplo, elementos armazenadores tipo

FIFO, FILO etc. Com isto, contempla-se a possibilidade de reutilizar estas definições em vários modelos, assim como expandir de forma fácil estes conceitos através do mecanismo da herança.

Também se pode utilizar os atributos dos objetos da rede GHENeSys para introduzir o conceito de tempo (discreto).

A rede GHENeSys foi definida no item anterior como uma quíntupla **N=(L,A,F,K,M)**, conforme a definição 1. Os conjuntos **L** e **A** são representados por as classes *Placê* e *Activity* respetivamente. Portanto, os elementos do conjunto **L** são objetos da classe *Place* e os elementos do conjunto **A** são objetos da classe *Activity*. Como num sistema de classes cada um destes objetos herda as propriedades e métodos da classe a que pertencem. Em GHENeSys tanto a classe *Place* quanto a classe *Activity* possuem subclasses que permitem representar diferentes elementos, incluindo elementos compostos.

Sendo assim a rede GHENeSys, é um agregado (um sistemas formado por objetos de diferentes classes) e não uma superclasse como em outras propostas. Nesta abordagem, também existe uma unificação das tendências "Objetos dentro de uma rede de Petri" e "Redes de Petri dentro de objetos" mais sem afetar o formalismo de redes. Nesta proposta a rede está composta por objetos pertencentes a duas classes e cada um deles está relacionado com objetos da classe oposta, mantendo assim a característica de bipartida das redes de Petri. Estes objetos por sua vez podem conter subredes sempre que estas estejam conforme a definição de elemento macro. Observando estas restrições o formalismo das redes é

conservado.

Outra vantagem da orientação a objetos que é aproveitada nesta rede é a possibilidade de colocar nos métodos comportamentos de elementos conhecidos ou já tratados em outros modelos o que caracteriza a reutilização. Isto pode ser feito sempre que a estrutura e a regra de disparo da rede sejam respeitados. Em principio isto pode ser visto como uma limitação, e de fato é, mas esta limitação existe para garantir a consistência do formalismo.

Apesar desta limitação, a qual consideramos indispensável, a introdução destes elementos constitui um avanço, pois permite encapsular e reaproveitar elementos e situações que normalmente aparecem nos modelos e que por outra parte atenuam o problema da explosão de estados, problema bem comum na modelagem de sistemas reais de mediano e grande porte.

Outra diferença fundamental nesta proposta com respeito as outras é o fato de que, para qualquer nível de abstração, temos a estrutura de uma rede a qual podemos aplicar todas as técnicas de análise conhecidas nas redes de Petri elementares.

Na figura 4.5 é mostrado o diagrama de classes proposto para nossa rede. Este diagrama pode ser ampliado segundo as necessidades de cada usuário dependendo da área de atuação e do sistema a modelar.

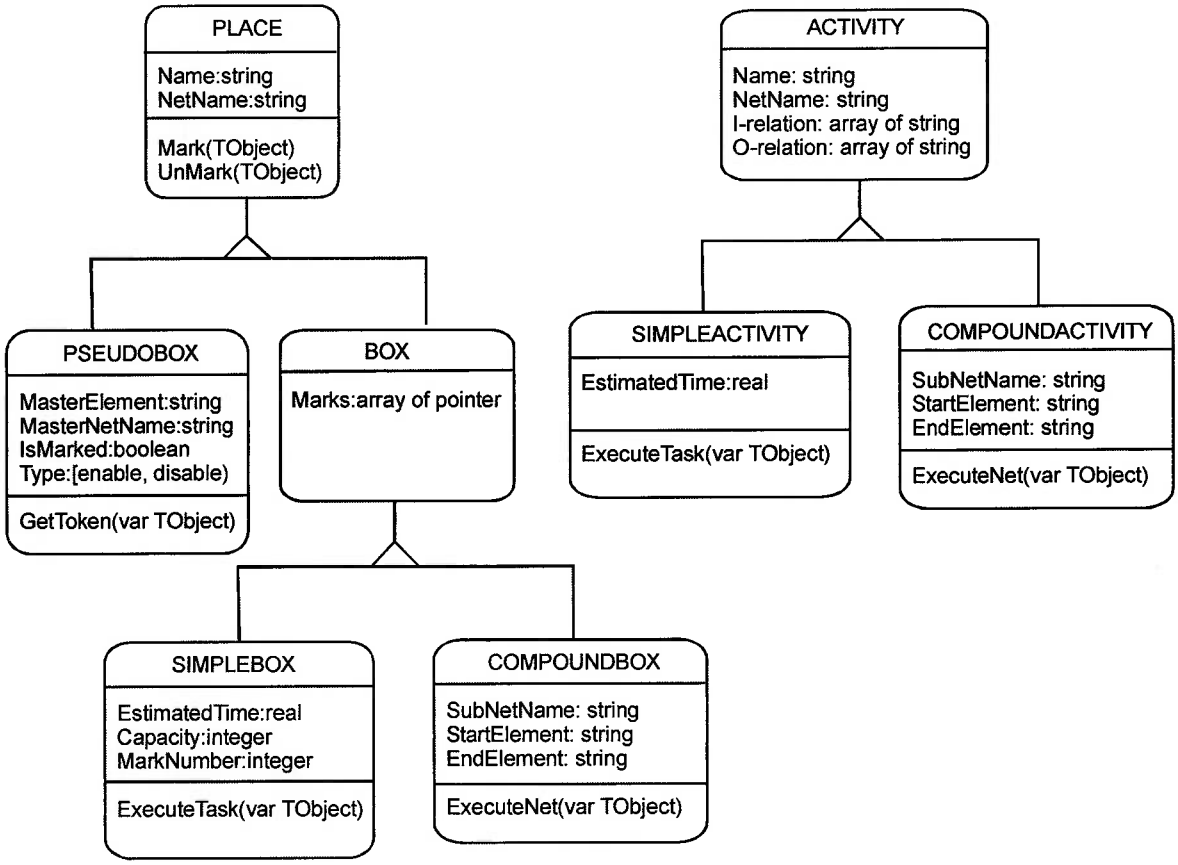


Fig. 4.5 Diagrama das classes *Place* e *Activity*.

Nesta estrutura de classes não foram colocados métodos para permitir a execução da regra de disparo da rede, isto deve-se ao fato de ser mais razoável fazer isto através da matriz de incidência e a regra de disparo tal como foram definidos no epígrafe anterior, pelo menos enquanto a marca continuar sendo interpretada como um elemento de tipo único.

A subclasse *PseudoBox* tem uma função fundamental na nossa rede. Como nos referimos no epígrafe anterior, os pseudobox são elementos da rede criados para modelar transferencia ou compartilhamento de dados entre módulos integrantes da rede ou para modelar informação externa (por exemplo intervenção dos usuários ou operadores no

sistema). Como não existe ligação através de arcos entre um pseudobox e o box que ele representa na rede (precisamente para facilitar a construção de modelos estruturados), esta ligação é feita através de mecanismos de orientação a objetos.

Os atributos MasterElement e MasterNetName da classe *PseudoBox* contém o nome do elemento e da subrede à qual pertence o elemento que o *pseudobox* representa, e serve para comunicar ao método **getToken** da mesma classe, que elemento na rede ele deve consultar para obter o estado atual do *pseudobo*x.

Em resumo, o sistema funciona da seguinte forma: primeiro é calculado o vetor de habilitação, a seguir determina-se se existem conflitos para determinar quais realmente serão as atividades a serem executadas. As atividades a serem executadas (dependendo da subclasse a que pertence: simples ou composta) executam os métodos **ExecuteTask** ou **ExecuteNet**.

Estes métodos através da informação colocada em i-relation disparam os métodos **getToken**, **ExecuteTask** ou **ExecuteNet** dependendo também da subclasse do elemento *Place*, que por sua vez devolvem um objeto (TObject) entre os alocados (através do atributo Marks), e remove o ponteiro se for o caso (exceto para os *pseudoBox*). As informações contidas dentro do objeto marca (TObject) é utilizado para realizar a tarefa programada no método e, para finalizar o disparo, é atualizado nos objetos da classe *Place*, relacionados no o-relation da atividade disparada, o atributo Marks, com os objetos resultantes da execução do método na atividade.

5 A ESTRUTURAÇÃO NA REDE GHENESYS.

Neste capítulo apresenta-se a análise de algumas das principais propriedades de redes que foram utilizadas na proposta do GHENeSys, tomando como base definições encontradas em artigos clássicos já consagrandos sobre redes de Petri [Murata, 89], [Reisig, 82], [Bernardinello, 92].

Além dos conceitos e propriedades fundamentais, outros "*features*" atuais serão introduzidos, tais como a hierarquização das redes. Através da hierarquia é possível dividir uma rede em partes, permitindo o refinamento de um nível abstrato até um ponto que nos permita obter um modelo do sistema, com um nível de detalhamento apropriado, e segundo as necessidades do processo de *design*. Porém, é muito importante que, a medida que refinamos o nosso modelo, os requerimentos associados às propriedades sejam mantidos, ou, no pior dos casos, afetados de forma intencional. É justamente por isso que uma análise detalhada do processo de refinamento e seu impacto nas propriedades das redes se faz necessária.

O fato das redes de Petri serem concebidas para a descrição, de forma uniforme e exata, do maior número possível de fenômenos relacionados com a transmissão e transformação da informação [Petri, 73], faz com que exista um número infinito de possíveis estruturas

de redes a considerar. A introdução da estruturação e dos processos de refinamento torna importante que as propriedades analisadas em um determinado nível de abstração se conservem nos respectivos refinamentos.

Neste capítulo procuramos contribuir neste sentido, revendo os processos de análise de propriedades e sua consrevação na estrutura hierárquica da rede GHENeSys.

5.1 Análise das propriedades na rede GHENeSys.

5.1.1 Invariantes de lugar.

Os invariantes de lugar constituem uma medida da conservação das marcas em uma rede. Na verdade obtem-se um vetor de ponderação de marcas (uma vez que nem todas as redes são conservativas) que multiplicado por qualquer marcação admissível da rede produz o mesmo resultado. Na rede GHENeSys os invariantes de lugar podem ser calculados resolvendo a seguinte equação: $\mathbf{C}^T.\mathbf{x} = \mathbf{0}$ onde **C** é a matriz de incidência, **0** é um vetor coluna com todos seus elementos iguais a 0. Todos os vetores coluna **x** que satisfazem esta equação, formam uma matriz que contém os invariantes de lugar da rede representada por **C**.

Para definir a relação entre os invariantes de lugar de uma rede GHENeSys com elemento macro, e a mesma rede com este elemento macro refinado colocamos o seguinte teorema.

TEOREMA 1. Seja **N** uma rede GHENeSys com ao menos um elemento macro representado pela matriz de incidência **C,** e $\mathcal{M}$ a mesma rede com este elemento refinado, representado pela matriz de incidência **B**.

Se **X** é a matriz cujas colunas **x** são soluções da equação $\mathbf{C}^T.\mathbf{x} = 0$, portanto os invariantes de lugar associados a **C**, e **Xr** é a matriz cujas colunas $\mathbf{x}_r$ são a solução da equação $\mathbf{B}^T.\mathbf{x}= 0$ então se cumpre que **X** é uma submatriz de **Xr**.

Dem] Para provar isto dividiremos nossa prova em duas partes, a primeira quando o elemento refinado é um box e a segunda quando o elemento refinado é uma atividade.

Seja **C** a matriz de incidência de **N** com *m* lugares e *n* atividades e um elemento macro no lugar m[4]. Seja **B** a matriz de incidência de $\mathcal{M}$ onde o lugar *m* (que é um *box*) foi refinado em uma subrede com *i* lugares e *j* atividades.

Temos os seguintes sistemas de equações:

---

[4] Escolhemos, sem perda de generalidade, o último lugar da rede como elemento a ser refinado de modo a obter uma matriz onde as submatrizes de expansão estão no canto superior direito. Esta forma é particularmente conveniente para o cálculo das propriedades, mas não implica em em nenhuma restrição dado que é sempre possível permutar linhas e colunas de modo a obtê-la, sem alterar as propriedades da matriz, salvo o sinal do determinante. Também consideramos que o elemento *m* é um *box* pois os *pseudoboxes* não podem ser refinados.

$$C_{11}x_1 + \ldots + C_{m1}x_m = 0$$

$$C_{1n}x_1 + \ldots + C_{mn}x_m = 0$$

$$\left.\begin{array}{l} B_{11}x_1 + \ldots + B_{(m-1)1}x_{(m-1)} + B_{m1}x_{(m)} + \ldots + B_{(m-1+i)1}x_{(m-1+i)} = 0 \\ \\ B_{1n}x_1 + \ldots + B_{(m-1)n}x_{(m-1)} + B_{mn}x_{m+1} + \ldots + B_{(m-1+i)n}x_{(m-1+i)} = 0 \end{array}\right\} \text{primeiras n equações}$$

$$\left.\begin{array}{l} B_{1(n+1)}x_1 + \ldots + B_{(m-1)(n+1)}x_{(m-1)} + B_{(m)(n+1)}x_{(m)} + \ldots + B_{(m-1+i)(n+1)}x_{(m-1+i)} = 0 \\ \\ B_{1(n+j)}x_1 + \ldots + B_{(m-1)(n+j)}x_{(m-1)} + B_{(m)(n+j)}x_{(m)} + \ldots + B_{(m-1+i)(n+j)}x_{(m-1+i)} = 0 \end{array}\right\} \begin{array}{l} \text{últimas j} \\ \text{equações} \end{array}$$

Pela definição de elemento macro os elementos da nova subrede não têm nenhuma relação com os elementos da rede original exceto os elementos de entrada e saída que assumem as relações do elemento macro portanto para m-1+i>y>m z>=n Byz=0 para as primeiras n equações e para y<m-1 , z<n+1 Byz=0 para as últimas j equações :

$$\left.\begin{array}{l} B_{11}x_1 + \ldots + B_{m1}x_m + B_{(m-1+i)1}x_{(m-1+i)} = 0 \\ \\ B_{11}x_n + \ldots + B_{mn}x_m + B_{(m-1+i)n}x_{(m-1+i)} = 0 \end{array}\right\} \text{primeiras n equações}$$

$$B_{m(n+1)}x_m + \ldots + B_{(m-1+i)(n+1)}x_{(m-1+i)} = 0$$

$$B_{m(n+j)}x_m + \ldots + B_{(m-1+i)(n+j)}x_{(m-1+i)} = 0$$

Os elementos da matriz **B** presentes nas primeiras n equações exceto os coeficientes dos termos x_m e $x_{(m-1+i)}$ são os mesmos da matriz **C** e estes não aparecem nas demais equações.

Para os coeficientes dos termos x_m e $x_{(m-1+i)}$ que são os que representam o ponto de entrada e saída respetivamente do elemento macro cumpre-se que:

$$B_{m1}x_m + B_{(m-1+i)1}x_{(m-1+i)} = C_{m1}x_m$$

$$B_{mn}x_m + B_{(m-1+i)n}x_{(m-1+i)} = C_{mn}x_m$$

Isto prova que, as soluções do primeiro sistema de equações estão contidas nas soluções do segundo sistema de equações. □

Agora precisamos provar que o mesmo acontece se refinamos um elemento atividade. Se refinamos o elemento n com uma rede com *i* lugares e *j* atividades teremos o seguinte sistema de equações:

$$B_{11}x_1 + \ldots + B_{m1}x_m + \ldots + B_{(m+i)1}x_{(m+i)} = 0$$

$$B_{1(n-1)}x_1 + \ldots + B_{m(n-1)}x_m + \ldots + B_{(m+i)(n-1)}x_{(m+i)} = 0$$

} primeiras n-1 equações

$$B_{1n}x_1 + \ldots + B_{mn}x_m + \ldots + B_{(m+i)n}x_{(m+i)} = 0$$

$$B_{1(n-1+j)}x_1 + \ldots + B_{m(n-1+j)}x_m + \ldots + B_{(m+i)(n-1+j)}x_{(m+i)} = 0$$

} últimas j equações

Novamente pela definição de elemento macro os elementos da nova subrede não têm nenhuma relação com os elementos da rede exceto os elementos de entrada e saída que assumem as relações do elemento macro portanto para m>y z>n Byz=0 para as primeiras n-1 equações e para y<m , z=<n Byz=0 para as últimas j equações :

$$B_{11}x_1 + \ldots + B_{m1}x_m = 0$$

$$B_{1(n-1)}x_1 + \ldots + B_{m(n-1)}x_m = 0$$

} primeiras n-1 equações

$$B_{1n}x_1 + \ldots + B_{mn}x_m + \ldots + B_{(m+i)n}x_{(m+i)} = 0$$

$$B_{(m+1)(n+1)}x_{(m+1)} + \ldots + B_{(m+i)(n+1)}x_{(m+i)} = 0$$

$$B_{(m+1)(n-2+j)}x_m + \ldots + B_{(m+i)(n-2+j)}x_{(m+i)} = 0$$

$$B_{1(n-1+j)}x_1 + \ldots + B_{m(n-1+j)}x_m + \ldots + B_{(m+i)(n-1+j)}x_{(m+i)} = 0$$

} últimas j +1 equações

As primeiras n-1 equações correspondentes a $\mathbf{N}_r$ são iguais as primeiras n-1 equações correspondentes a **N** e tirando a primeira e última equação das últimas j+1 equações vemos que não existe nenhuma relação entre as variáveis de uma com a outra portanto as soluções são independentes. As duas equações restantes podem ser arranjadas da seguinte forma:

$$B_{1n}x_1 + \ldots + B_{(m+i)n}x_{(m+i)} + B_{1(n-1+j)}x_1 + \ldots + B_{(m+i)(n-1+j)}x_{(m+i)} = 0$$

e logo:

$$(B_{1n} + B_{1(n-1+j)})x_1 + \ldots + (B_{(m+i)n} + B_{(m+i)(n-1+j)})x_{(m+i)} = 0$$

esta equação pode ser dividida em duas, ficando

$$(B_{1n} + B_{1(n-1+j)})x_1 + \ldots + (B_{mn} + B_{m(n-1+j)})x_m = 0 \longrightarrow \text{equação 1}$$

$$(B_{(m+1)n} + B_{(m+1)(n-1+j)})x_{(m+1)} + \ldots + (B_{(m+i)n} + B_{(m+i)(n-1+j)})x_{(m+i)} = 0$$

Como os coeficientes dos termos da equação 1 são a soma dos elementos primeiro e último da subrede $\mathcal{M}$ para os m primeiros lugares, cumprem que:

$$(B_{1n} + B_{1(n-1+j)})x_1 + \ldots + (B_{mn} + B_{m(n-1+j)})x_m = C_{1n}x_1 + \ldots + C_{mn}x_m = 0$$

Portanto, as soluções do primeiro sistema de equações estão incluídas nas soluções do segundo □

Com este teorema podemos garantir que no refinamento de elementos numa rede GHENeSys, sempre que a subrede colocada como refinamento tenha só um ponto de

entrada e um de saída, os invariantes de lugar da rede sem refinamento não serão afetados.

5.1.2 Invariantes de atividade.

Os invariantes de atividade nos permitem conhecer o número de vezes que um determinado conjunto de atividades devem ser executadas para o sistema voltar a seu estado inicial. Na rede GHENeSys os invariantes de atividade podem ser calculadas resolvendo a seguinte equação: $\mathbf{C.x = 0}$ onde **C** é a matriz de incidência, **0** é um vetor coluna com todos seus elementos iguais a 0. Todos os vetores coluna **x** que satisfazem esta equação são as invariantes de atividade da rede representada por **C**.

Similarmente ao que fizemos com as invariantes de lugar, vamos definir a influência do refinamento de elementos macro na preservação dos invariantes de atividade.

TEOREMA 2. Seja **N** uma rede GHENeSys com ao menos um elemento macro representado pela matriz de incidência **C** e $\mathbf{N_r}$ a mesma rede com este elemento refinado representada pela matriz de incidência **B**. Se **X** é a matriz cujas colunas **x** são a solução da equação $\mathbf{C.x = 0}$. Se **Xr** é a matriz cujas colunas **x** são a solução da equação $\mathbf{B.x = 0}$, então **X** é uma submatriz de **Xr**.

Dem] Seguindo o mesmo procedimento adotado na demonstração anterior dividiremos esta em duas partes, a primeira quando o elemento refinado é um lugar e a segunda quando o elemento refinado é uma atividade.

Seja **C** a matriz de incidência de **N** com m lugares e n atividades e um elemento macro na

atividade *n*. Seja **B** a matriz de incidência de **N**r onde a atividade *n* foi refinada em uma rede com *i* lugares e *j* atividades.

Temos os seguintes sistemas de equações:

$$C_{11}x_1 + \ldots + C_{1n}x_n = 0$$

$$C_{m1}x_1 + \ldots + C_{mn}x_n = 0$$

$$\left.\begin{array}{l} B_{11}x_1 + \ldots + B_{1(n-1)}x_{(n-1)} + B_{1n}x_n + \ldots + B_{1(n-1+j)}x_{(n-1+j)} = 0 \\ \\ B_{m1}x_1 + \ldots + B_{m(n-1)}x_{(n-1)} + B_{mn}x_n + \ldots + B_{m(n-1+j)}x_{(n-1+j)} = 0 \end{array}\right\} \text{primeiras } m \text{ equações}$$

$$\left.\begin{array}{l} B_{(m+1)1}x_1 + \ldots + B_{(m+1)(n-1)}x_{(n-1)} + B_{(m+1)n}x_n + \ldots + B_{(m+1)(n-1+j)}x_{(n-1+j)} = 0 \\ \\ B_{(m+i)1}x_1 + \ldots + B_{(m+i)(n-1)}x_{(n-1)} + B_{(m+i)(n)}x_n + \ldots + B_{(m+i)(n-1+j)}x_{(n-1+j)} = 0 \end{array}\right\} \begin{array}{l} \text{últimas } i \\ \text{equações} \end{array}$$

Pela definição de elemento macro os elementos da nova subrede não têm nenhuma relação com os elementos da rede exceto os elementos de entrada e saída que assumem as relações do elemento macro portanto para y>=*m*, *n*-1+j>z>*n* Byz=0 para as primeiras *m* equações e para y<=*m* , z<*n*-1 Byz=0 para as últimas *i* equações :

$$B_{11}x_1 + \ldots + B_{1(n-1)}x_{(n-1)} + B_{1n}x_n + B_{1(n-1+j)}x_{(n-1+j)} = 0$$

$$B_{m1}x_1 + \ldots + B_{m(n-1)}x_{(n-1)} + B_{mn}x_n + B_{m(n-1+j)}x_{(n-1+j)} = 0$$

} primeiras *m* equações

$$B_{(m+1)n}x_n + \ldots + B_{(m+1)(n-1+j)}x_{(n-1+j)} = 0$$

$$B_{(m+i)(n)}x_n + \ldots + B_{(m+i)(n-1+j)}x_{(n-1+j)} = 0$$

Pela mesma razão citada na prova anterior têm-se que:

$$B_{1n}x_n + B_{1(n-1+j)}x_{(n-1+j)} = C_{1n}x_n$$

$$B_{mn}x_n + B_{m(n-1+j)}x_{(n-1+j)} = C_{mn}x_n$$

Isto prova que as soluções do primeiro sistema de equações estão contidas nas soluções do segundo sistema de equações. □

Agora precisamos provar que o mesmo acontece se refinamos um elemento atividade. Se refinamos o elemento n com uma rede com *i* lugares e *j* atividades teremos o seguinte sistema de equações:

$$B_{11}x_1 + \ldots + B_{1n}x_n + \ldots + B_{1(n+j)}x_{(n+j)} = 0$$

$$B_{(m-1)1}x_1 + \ldots + B_{(m-1)n}x_n + \ldots + B_{(m-1)(n+j)}x_{(n+j)} = 0$$

} primeiras m-1 equações

$$B_{m1}x_1 + \ldots + B_{mn}x_n + \ldots + B_{m(n+j)}x_{(n+j)} = 0$$

$$B_{(m+i)1}x_1 + \ldots + B_{(m+i)n}x_n + \ldots + B_{(m+i)(n+j)}x_{(n+j)} = 0$$

} últimas i equações

Pela definição de elemento macro temos que: para m>=y, z>n Byz=0 para as primeiras m-1 equações e para y<m , z=<n Byz=0 para as últimas j equações :

$$B_{11}x_1 + \ldots + B_{1n}x_n = 0$$

$$B_{(m-1)1}x_1 + \ldots + B_{(m-1)n}x_n = 0$$

} primeiras m-1 equações

$$B_{m1}x_1 + \ldots + B_{m(n+j)}x_{(n+j)} = 0$$

$$B_{(m+1)n}x_n + \ldots + B_{(m+1)(n+j)}x_{(n+j)} = 0$$

$$B_{(m-1+i)n}x_n + \ldots + B_{(m-1+i)(n+j)}x_{(n+j)} = 0$$

$$B_{(m+i)1}x_1 + \ldots + B_{(m+i)(n+j)}x_{(n+j)} = 0$$

} últimas i+1 equações

As primeiras *m*-1 equações correspondentes a N_r são iguais as primeiras *m*-1 equações correspondentes a N e tirando a primeira e última equação das ultimas *i*+1 equações vemos que não existe nenhuma relação entre os variáveis de uma com a outra portanto as soluções são independentes. As duas equações restantes podem ser arranjadas da seguinte forma:

$$B_{m1}x_1 + \ldots + B_{m(n+j)}x_{(n+j)} + B_{(m+i)1}x_1 + \ldots + B_{(m+i)(n+j)}x_{(n+j)} = 0$$

e logo:

$$(B_{m1} + B_{(m+i)1})x_1 + \ldots + (B_{m(n+j)} + B_{(m+i)(n+j)})x_{(n+j)} = 0$$

esta equação pode ser dividida em duas, ficando:

$$(B_{m1} + B_{(m+i)1})x_1 + \ldots + (B_{mn} + B_{(m+i)n})x_n = 0 \longrightarrow \text{equação 1}$$

$$(B_{m(n+1)} + B_{(m+i)(n+1)})x_{(n+1)} + \ldots + (B_{m(n+j)} + B_{(m+i)(n+j)})x_{(n+j)} = 0$$

Como os coeficientes dos termos da equação 1 são a soma dos elementos primeiro e último da subrede m para os *m* primeiros lugares cumpre-se que:

$$(B_{m1} + B_{(m+i)1})x_1 + \ldots + (B_{mn} + B_{(m+i)n})x_n = C_{m1}x_1 + \ldots + C_{mn}x_n = 0$$

Portanto as soluções do primeiro sistema de equações estão incluídos nas soluções do segundo □

Com este teorema podemos garantir que no refinamento de elementos numa rede GHENeSys, sempre que a subrede colocada como refinamento tenha só um ponto de

entrada e um de saída, os invariantes de atividade da rede sem refinamento não serão afetados.

5.1.3 Vivacidade.

A vivacidade é uma propriedade ligada a ausência de bloqueios nos sistemas que a rede representa. Por causa disso é muito importante conhecer a influência dos refinamentos na preservação desta propriedade. Levando em consideração as definições do capítulo anterior apresentamos o seguinte teorema:

TEOREMA 3. Seja uma rede $\mathbf{N}=(\mathbf{L},\mathbf{A},\mathbf{F},\mathbf{K},\mathbf{M}_0)$ com um elemento macro, e vivacidade definida como Li-viva, uma rede $\mathbf{Nr}=(\mathbf{Lr},\mathbf{Ar},\mathbf{Fr},\mathbf{Kr},\mathbf{Mr})$ onde o elemento macro foi substituído pela rede **Nm**, então, a vivacidade da rede **Nr** será igualmente Li-viva se e somente se a vivacidade de **Nm** com marcação inicial **Mm** é Lj-viva com $j \geq i$.

Dem] Se o elemento a ser refinado é um box, que denominaremos b_e onde $b_e \in \mathbf{B}$ então:

Como a rede não tem elementos desconectados, $\exists |\ a \in \mathbf{A}$, $a \in {}^\bullet b_e$ em **N**. Vamos supor que a rede **N** é Li-viva e a subrede **Nm** é Lj-viva, com $i \neq j$. Para toda seqüência de disparo $\sigma \in L(\mathbf{M}_0)$ a troca do lugar b_e pela subrede **Nm** não altera a vivacidade porque **Nm** é uma subrede própria e portanto tem se uma seqüência que atinge a saída da subrede própria acrescentando-se uma seqüência de eventos que não pertencem a **N**.

Para os eventos de L(**Mr**), se $j \geq i$, então, estes serão Li-vivos, dado que só pertencem às

seqüências $\sigma \in L(\mathbf{M}_0)$ que habilitam o evento $a \in {}^{\bullet}b_e$. Se j < i, teremos um conjunto de seqüências $\sigma_m \in L(\mathbf{M}_m)$ que restringe as propriedades anteriormente atribuídas a $\sigma \in L(\mathbf{M}_0)$, e pela definição 11, $\sigma_r \in L(\mathbf{M}_r)$ é Lj-viva.

Se o elemento a ser refinado é uma atividade a demonstração é similar, onde a atividade *a* usada na demonstração é a própria atividade macro que é substituída por uma seqüência de atividades onde os elementos extremos são as atividades a_{el} e a_{ej} que limitam a subrede própria.

Evidentemente se o elemento macro atende a definição 17 então é um elemento macro que denominaremos "bem comportado", quer dizer L4, que é o maior nível de vivacidade, e portanto como $4 \geq i$ a vivacidade da rede, dependerá unicamente da vivacidade da rede no nível mais abstrato. Se garantimos que a rede no nível mais abstrato é viva então refinando os elementos macro com blocos "bem comportados" a rede resultante sempre será viva.

5.1.4 Equidade.

A propriedade de equidade é uma medida da diferença no número de ocorrências entre duas atividades de uma rede. Como existem dois conceitos básicos para esta propriedade escolhemos a definição 14 para determinar a influência dos refinamentos na manutenção desta propriedade.

TEOREMA 4. Seja uma rede $\mathbf{N}=(\mathbf{L},\mathbf{A},\mathbf{F},\mathbf{K},\mathbf{M}_0)$ com um elemento macro, e incondicionalmente eqüitativa, uma rede $\mathbf{Nr}=(\mathbf{Lr},\mathbf{Ar},\mathbf{Fr},\mathbf{Kr},\mathbf{Mr})$ onde o elemento macro foi

substituído pela rede **N**m, então, a rede **N**r será incondicionalmente eqüitativa se e somente se a rede **N**m com marcação inicial **M**m for incondicionalmente eqüitativa.

Dem] Se **N** é incondicionalmente eqüitativa significa que todas as seqüências de disparo σ em R(**M**0) são incondicionalmente eqüitativas. Concentraremos esta prova nas redes cíclicas pois as redes finitas e não cíclicas são por definição incondicionalmente eqüitativas.

Como **N** é incondicionalmente eqüitativa todas as atividades $a \in \mathbf{A}$ aparecem com freqüência infinita em σ. Se o elemento macro a ser substituído é um lugar, que denominaremos b_e onde $b_e \in \mathbf{L}$ então:

Seja uma atividade $a_e \in \mathbf{A}$ tal que $a_e \in {}^{\bullet}b_e$. Sejam $\sigma_e \subseteq L(\mathbf{M}_0)$ as seqüências tais incluem a_e. Após a substituição de b_e pela subrede **N**m estas seqüências aumentarão em uma seqüência composta exclusivamente das atividades pertencentes a **N**m (uma vez que o **N**m é uma subrede própria). Portanto, como estas seqüências são finitas, as atividades de **N** continuam aparecendo com freqüência infinita e são incondicionalmente eqüitativas entre si. Porém se **N**m não for incondicionalmente eqüitativa as seqüências $\sigma \in L(\mathbf{Mr})$ não mais serão incondicionalmente eqüitativas, quer seja porque as atividades de **N**m não são eqüitativas entre si ou porque não são incondicionalmente eqüitativas com as atividades da rede **N**.

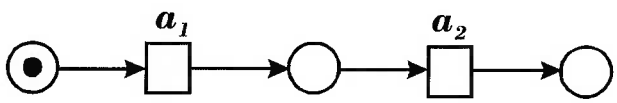
Se o elemento a ser substituído é uma atividade o argumento é o mesmo, só que usando a própria atividade ao invés de um elemento do conjunto dos seus predecessores.

5.2 Estruturas relacionadas ao controle de processos.

Uma das partes fundamentais no controle de processos é gerenciar tarefas a serem realizadas para completar a produção de um ou vários produtos, ou a realização de um serviço. Estes mecanismos de controle podem ser representados através de redes de Petri ou similares. Existem algumas estruturas básicas de redes de Petri que representam estes comportamentos. Estas estruturas são freqüentemente usadas na construção de modelos e podem ser divididas em dois grandes grupos: as equivalentes as estruturas de controle em linguagens de programação, e as estruturas básicas em sistemas de produção.

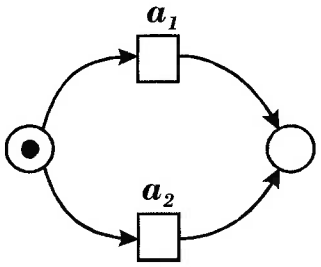
O primeiro grupo está formado pelas seguintes estruturas:

- **Seqüencial**: As tarefas são executadas de forma seqüencial uma depois da outra. Esta estrutura é fundamental pois constitui a base de construção de qualquer modelo ou programa (ou de sistema de produção simples).



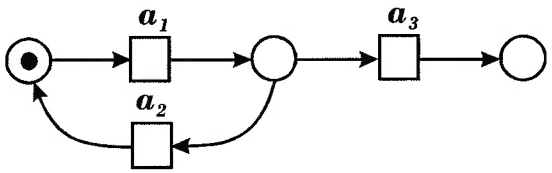
a tarefa a_2 é executada uma vez concluída a tarefa a_1.

- **Condicional**: Numa determinada situação é preciso escolher uma tarefa entre várias para ser executada.



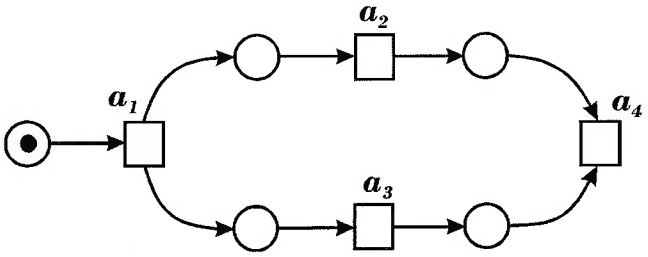
uma das tarefas a_1 ou a_2 é executada.

- **Iteração**: Uma mesma tarefa é executada várias vezes.



As tarefas a_1 ou a_2 podem ser executadas várias vezes antes que a tarefa a_3 seja executada.

- **Paralela**: Duas tarefas são executadas ao mesmo tempo em paralelo.

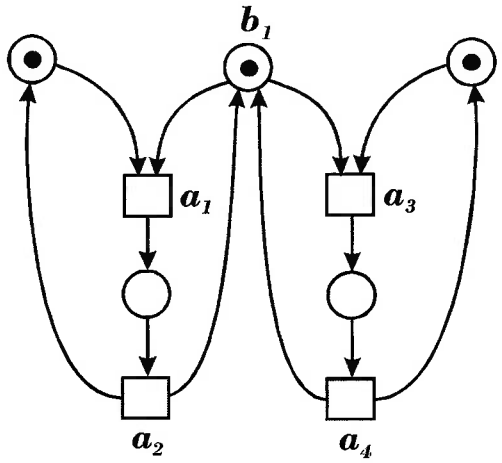


As tarefas a_2 e a_3 são executadas em paralelo.

As estruturas do primeiro grupo são todas estruturas próprias. Isto resulta evidente pois, representam os elementos primos que são a base de construção dos programas estruturados.

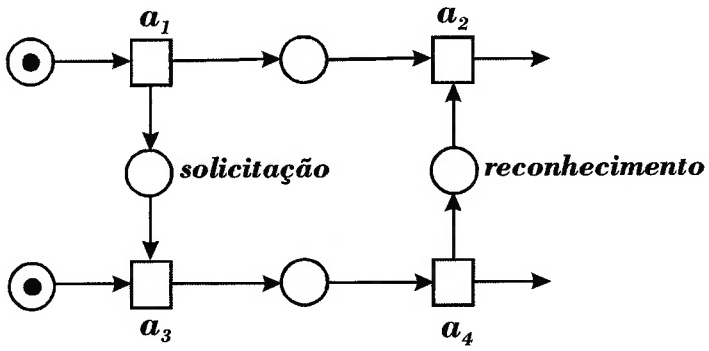
O segundo grupo está formado, entre outras, pelas seguintes estruturas:

- **Compartilhamento de recursos (informação)**: é o caso onde dois processos paralelos compartilham um determinado recurso.



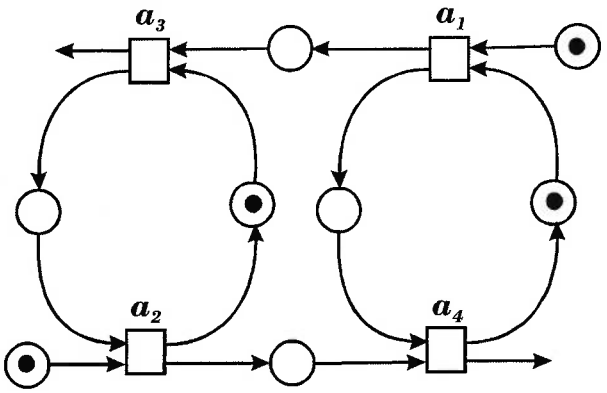
As tarefas a_1 e a_3 compartilham o recurso em b_1, uma vez concluídas as tarefas a_2 ou a_4 o recurso é disponibilizado novamente.

- **Sincronização**: Dois processos são sincronizados através de uma relação pedido/reconhecimento.



As tarefas a_1 e a_2.estão relacionadas com a execução de a_3 e a_4

- **Relação produtor-consumidor**: É a relação de execução mútua de duas seções específicas de um processo onde a execução de um deles habilita a execução do outro e vice-versa.



A execução das tarefas *a1* e *a3*,leva a execução das tarefas *a2* e *a4* que por sua vez levam novamente a execução de *a1* e *a3* e assim por diante.

Estas estruturas são as que aparecem freqüentemente em Sistemas Flexíveis de Manufatura (SFM) [Hasegawa, 96].

As estruturas do segundo grupo não são estruturas próprias pois, possuem mais de uma entrada ou mais de uma saída.

Se procuramos construir modelos estruturados devemos tentar usar, se não totalmente, ao menos em sua maioria as estruturas do primeiro grupo para compor o modelo. Nem sempre isto é possível, quando se trata de SFM, as estruturas do segundo grupo terão de ser usadas.

Agora, dependendo da situação, as estruturas do segundo grupo podem ser modificadas e transformadas também em estruturas próprias. No caso do compartilhamento de informação (não de recursos) onde, devido à existência dos *pseudoboxes* na rede GHENeSys, é possível representar este fato usando estruturas similares as do primeiro grupo. O mesmo pode acontecer com a sincronização, baseada na transferência de sinais e/ou dados, que se "encaixam" perfeitamente na semântica dos *pseudoboxes*.

Já o caso do compartilhamento de recursos não é trivial, o fato dos *pseudoboxes* serem elementos de marcação permanente não permite modelar recursos que são consumidos ou utilizados por vários equipamentos ou processos.

Nestes casos onde não é possível usar ou adaptar estruturas próprias, a solução seria modelar esta parte do sistema dentro de um bloco com uma entrada e uma saída de forma que possamos isolar estas "fontes de bloqueios" e prever algoritmos de controle para resolver estes problemas.

No caso de existência de comportamentos inadequados teremos devidamente identificado onde procurar e resolver estes problemas. Para sistemas de grande porte, isto consiste em um fator de auxílio no processo de modelagem.

5.3 Aplicação das redes GHENeSys a modelagem e análise de workflows.

A maioria das ferramentas para gerenciamento de *workflow* não são baseadas nas redes de Petri. Mesmo assim existem algumas ferramentas para análise e *design* de *workflows* baseadas em redes, como por exemplo o COSA(Software Ley/COSA Solutions, Pullheim, Alemanha), INCOME (Promatis, Karlsbad, Alemanha), e LEU (Vebacom, Bochum, Alemanha).

Segundo van der Aalst, o gerenciamento de *workflow* se tornará um domínio de aplicação

muito beneficiado pelas características que as redes de Petri oferecem [van der Aalst, 97]. Existem pelo menos três boas razões para se utilizar redes de Petri para a modelagem e análise de *workflow*: i) as redes possuem uma semântica formal, além de uma natureza gráfica suficiente para representar *workflows* primitivos identificados pela WFMC (Workflow Management Coalition), ii) a possibilidade de utilizar uma abordagem baseada em estados no lugar da abordagem baseada em eventos, mais comunmente utilizadas até hoje, e finalmente iii) a disponibilidade de várias técnicas de análise.

Nesta seção pretendemos mostrar através de um exemplo o potencial das redes GHENeSys para a modelagem e análise de *workflow*.

O exemplo em questão, tirado do artigo do próprio van der Aalst, é a modelagem de processamento de reclamações. A rede de Petri para modelar este processo no artigo de van der Aalst é mostrada na figura 5.1.

O processamento das reclamações é feito da seguinte forma: Primeiramente a reclamação é registrada, em seguida e de modo paralelo um questionário é enviado a quem fez a reclamação, ao mesmo tempo que a reclamação é avaliada. Se o reclamante retorna o questionário em duas semanas ou menos o mesmo será processado, senão o questionário é descartado. Baseado no resultado da avaliação a reclamação é processada ou não. O processamento da reclamação só é feito se o questionário foi processado ou se o tempo de espera foi esgotado. O processamento da reclamação é verificado e se não há falha a corrigir a reclamação é arquivada.

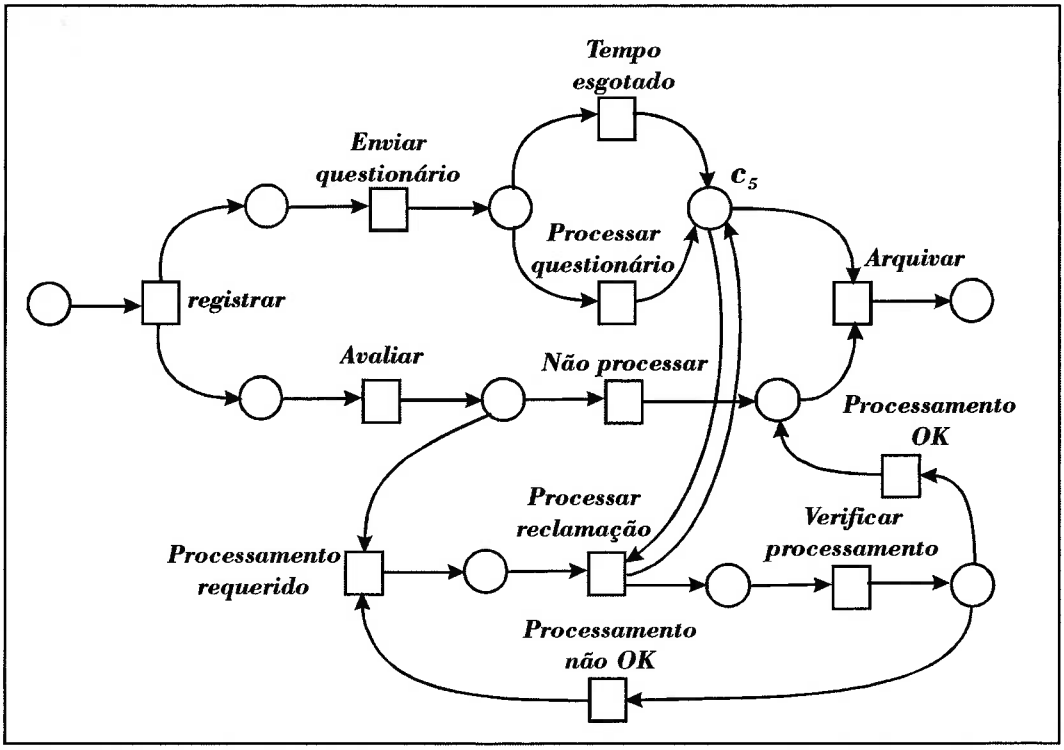


Fig 5.1 Modelagem do processamento de reclamações usando redes de Petri.

O exemplo em questão, tirado do artigo do próprio van der Aalst, é a modelagem de processamento de reclamações. A rede de Petri para modelar este processo no artigo de van der Aalst é mostrada na figura 5.1.

O processamento das reclamações é feito da seguinte forma: Primeiramente a reclamação é registrada, em seguida e de modo paralelo um questionário é enviado a quem fez a reclamação, ao mesmo tempo que a reclamação é avaliada. Se o reclamante retorna o questionário em duas semanas ou menos o mesmo será processado, senão o questionário é descartado. Baseado no resultado da avaliação a reclamação é processada ou não. O processamento da reclamação só é feito se o questionário foi processado ou se o tempo de espera foi esgotado. O processamento da reclamação é verificado e se não há falha a

corrigir a reclamação é arquivada.

Quando examinamos as representações em redes de Petri [van der Alst 97] e no GHENeSys, podemos confirmar que ambos possuem exatamente o mesmo comportamento (são funcionalmente equivalentes), embora seja mais difícil o processo de análise no modelo acima devido a existência de um laço (*self loop*) no lugar c5. Além disso, este tipo de representação gráfica não é representável na matriz de incidência e portanto a maioria das técnicas de análise de redes de Petri não poderiam ser aplicadas. Se aplicássemos as regras para obter um modelo alternativo estruturado, poderíamos corrigir este problema.

Nas figuras 5.2, 5.3 e 5.4 mostra-se como é gerado o modelo com a rede GHENeSys.

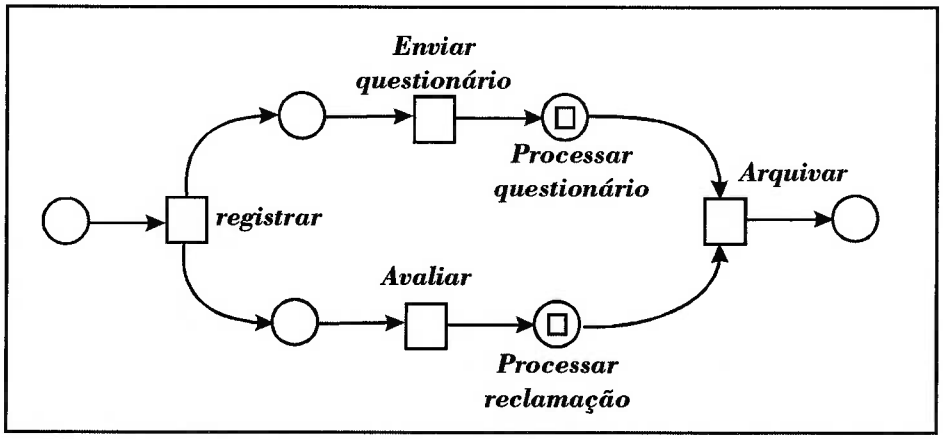


Fig. 5.2 Modelagem do processamento de reclamações no seu nível mais abstrato com a rede GHENeSys.

Este modelo é obtido a partir de refinamentos sucessivos. Note-se que em cada um dos diferentes níveis de abstração, representados aqui em figuras diferentes, temos uma rede perfeitamente válida à qual podemos aplicar todas as técnicas de análises vistas anteriormente.

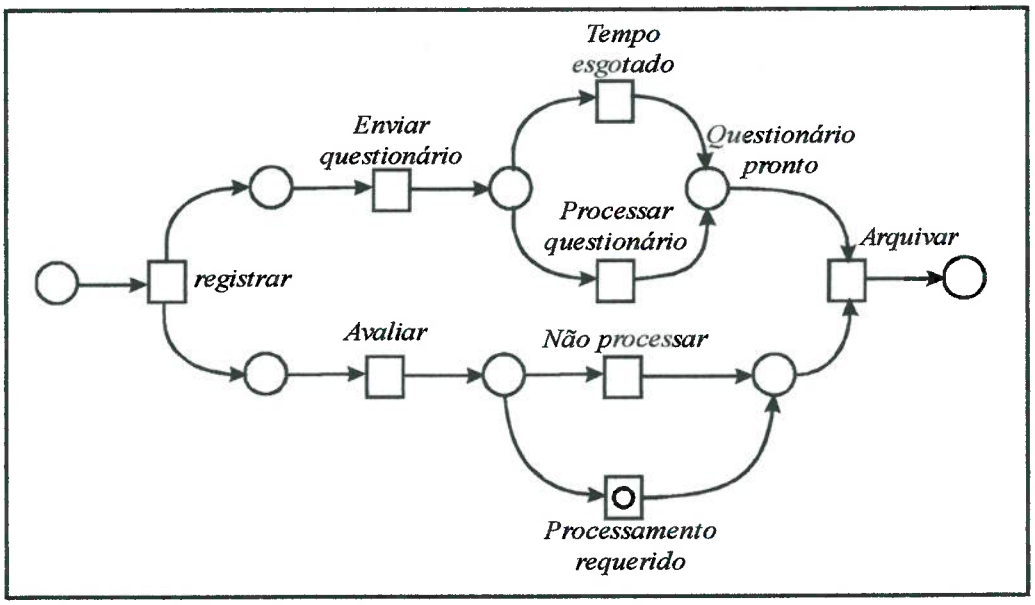


Fig. 5.3 Primeiro refinamento do modelo do processamento de reclamações.

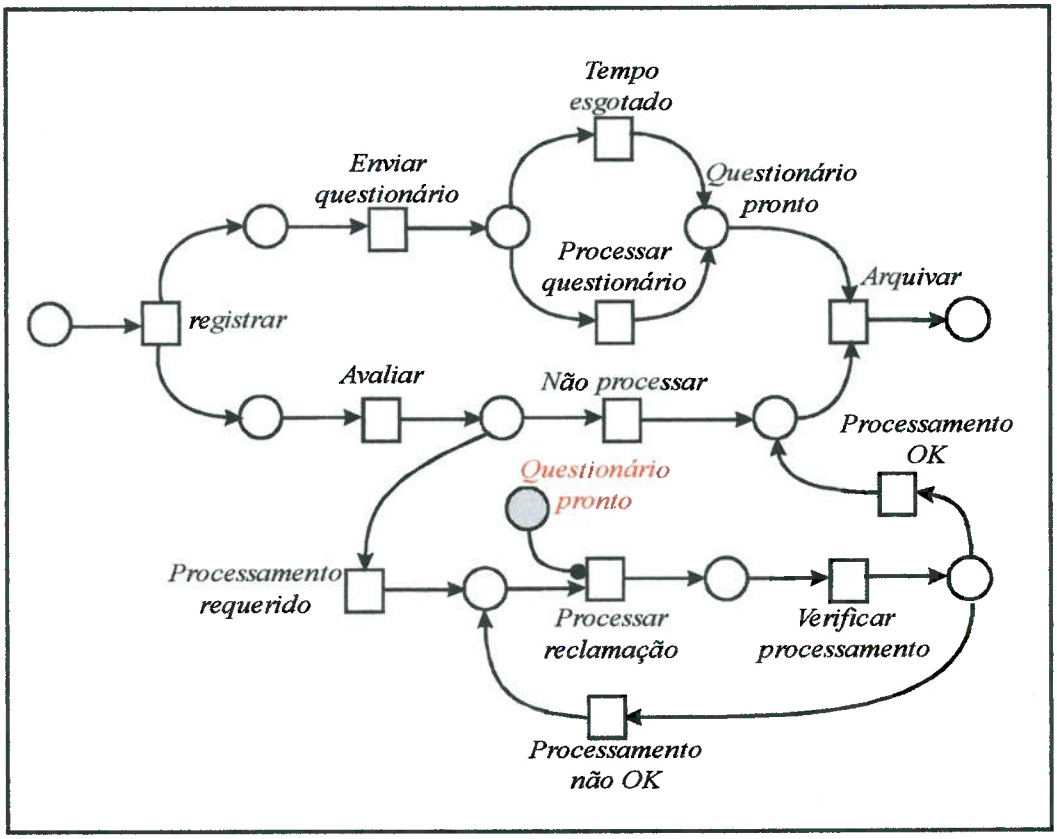


Fig. 5.4 Versão final do modelo do processamento de reclamações.

Na figura 5.4, onde aparece o modelo em redes GHENeSys do processamento de reclamações, podemos observar a similitude deste com o modelo da figura 5.1. A diferença mais visível é a aparição do elemento *pseudobox* que representa a transferência da informação dos dois ramos da mesma rede.

Uma diferenças fundamental entre estes dois modelos é que, modelando usando GHENeSys, o modelo pode ser representado por uma matriz, segundo o exposto no capítulo 4 (não tem laços) ao contrario do exemplo da figura 5.1.

Para mostrar esta diferença procederemos a calcular as invariantes de lugar correspondentes a rede mostrada na figura 5.4. A seguir apresentamos a matriz de incidência da rede.

$$C = \begin{vmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \end{vmatrix}$$

As invariantes de lugar são mostradas a seguir:

$$x = \begin{vmatrix} 0 & 1 & 1 & 2 & \ldots \\ 0 & 0 & 1 & 1 & \ldots \\ 0 & 1 & 0 & 1 & \ldots \\ 0 & 0 & 1 & 1 & \ldots \\ 0 & 1 & 0 & 1 & \ldots \\ 0 & 0 & 1 & 1 & \ldots \\ 0 & 1 & 0 & 1 & \ldots \\ 0 & 1 & 1 & 2 & \ldots \\ 0 & 1 & 0 & 1 & \ldots \\ 0 & 1 & 0 & 1 & \ldots \\ 0 & 1 & 0 & 1 & \ldots \\ 0 & 0 & 0 & 0 & \ldots \end{vmatrix}$$

Outra diferença radica no fato de todos os elementos utilizados para construir o modelo da figura 5.4 podem ser facilmente identificados e comparados com estruturas de controle de linguagens de programação, e colocados de forma estruturada.

Os blocos que representam estas estruturas correspondem, com pequenas variações, aos identificados pela WFMC como as quatro construções de rotas básicas: seqüencial, paralelo, condicional e iterativo.

A definição de um processo de *workflow*, como o mostrado no exemplo anterior, especifica como um caso (de reclamação) é roteado, ou, também poderíamos dizer, que especifica a tarefa que deve ser executada e qual a ordem de execução. Mas isso não é tudo, além das tarefas, um processo de *workflow* também envolve recursos e trabalhadores que desempenham estas tarefas. Por exemplo, para realizar uma determinada tarefa um trabalhador deve estar disponível, assim como o recursos que serão utilizados para poder

cumpri-la.

Tudo isto indica que necessitamos colocar mais informações na rede para conseguir realmente modelar o processo de *workflow*. É justamente nesta parte que a rede GHENeSys pretende evoluir. Fazendo uso de conceitos da orientação a objetos, a marca, como objeto passivo, poderia carregar atributos que permitiriam representar diferentes tipos de trabalhadores e recursos. Além disso, as tarefas, representadas pelas atividades, poderiam ter também atributos que permitam identificar que tipo de trabalhador é apto para a tarefa representada pela atividade em questão, e que tipo de recurso é consumido e/ou gerado durante a execução da mesma.

Algo parecido com isto foi colocado numa extensão do MFG denominado Extended Mark Flow Graph ou E-MFG para simplificar [Santos Filho, 93]. Preparar a parte orientada a objetos da rede para fazer isto não seria muito complicado, o problema seria então que o comportamento da rede não representaria o comportamento e as relações entre estes objetos e justamente neste ponto que lançamos a seguinte questão: Vale a pena aumentar o poder de representação da rede a custa de sacrificar a sua base formal? Existem várias propostas onde a parte formal que garante a análise das propriedades é sacrificada (no E-MFG por exemplo) em troca da facilidade de síntese e de interpretação da rede.

5.4 Aplicação das redes GHENeSys à modelagem e análise de sistemas integrados e flexíveis.

Nos sistemas de grande porte desenvolvidos e construídos pelo homem convivem sistemas de diferentes tipos. Os sistemas relacionados com fenômenos físicos, cujo comportamento é modelado através de equações diferenciais ou funções de transferência. Este tipo de sistema é chamado de Sistema Dinâmico de Variáveis Continuas (SDVC). Estes sistemas são freqüentemente encontrados como parte de sistemas de controle com retroalimentação em equipamentos tais como servomecanismos, controle de temperatura, controle de nível e outros.

Existe outro tipo de sistema, totalmente diferente do anterior, em que os estados não são representados por valores contínuos mas por um conjunto de condições, e onde a mudança de estados no sistema depende da ocorrência de um determinado evento ou atividade. Estes sistemas são conhecidos como Sistemas Dinâmicos de Eventos Discretos (SDED).

Apesar dos sistemas de grande porte estarem formados por estes tipos de sistema, muitas vezes o processo de análise qualitativa dos sistemas é feito com base em uma representação (aproximação) a eventos discretos, que representa num nível de abstração adequado o comportamento. Outras vezes, esta simplificação não pode ser feita e uma abordagem híbrida é adotada.

Neste trabalho enfoca-se apenas a modelagem, análise e projeto de sistemas de eventos

discretos. Portanto, neste nível de representação abstrata, é preciso enfocar os conceitos recentemente introduzidos de integração e flexibilidade. Os sistemas atuais, notadamente os sistemas automatizados estão fortemente associados a ambos os conceitos, principalmente no que concerne à manufatura moderna.

Existem dois tipos de integração: a integração fraca, que está associada à definição de sistemas (no sentido de que todo sistema, principalmente os de grande porte, são compostos de subsistemas que têm algum tipo de integração entre si), e a integração forte onde os subsistemas, além da sua composição funcional fraca, admitem o compartilhamento de recursos, informações, ou dados [Silva, 98]. O principal interesse deste trabalho é na integração forte, e estaremos nos referindo a esta, mesmo quando usarmos simplesmente o termo integração.

A flexibilidade é definida pela capacidade de cada um dos subsistemas componentes de possuir várias funcionalidades, embora só possam utilizar uma por vez (como as máquinas de comando numérico que podem assumir funções de desbaste, corte, furação, dependendo da ferramenta e da programação). A configuração destes sistemas flexíveis pode mudar pela combinação das variadas funcionalidades de maneira integrada e de modo a produzir uma função global distinta [Silva, 98].

Um exemplo deste tipo de sistemas são os Sistemas Flexíveis de Manufatura (SFM). O comportamento destes sistemas possui características complicadas devido a estes apresentarem uma combinação de atividades e condições de recursos em diferentes formas:

seqüencial, concorrente, em conflito, sincronizadas; além de um fluxo complexo de entidades discretas [Hasegawa, 96].

As Redes de Petri são consideradas uma das melhores ferramentas para representar este tipo de sistema. Suas características, amplamente citadas na literatura, [Martinez, 86] garantem a sua adequação na análise, projeto e simulação de tais sistemas.

Como a rede GHENeSys é uma extensão das redes de Petri, mantendo a base formal, e a possibilidade de análise de propriedades através de métodos matemáticos, podemos concluir que esta possui, no mínimo, as mesmas características que fazem das redes de Petri uma ferramenta fundamental para o análise e projeto de SFM.

Se agregamos a estes recursos de modelagem e análise a representação em diferentes níveis de abstração através do uso da hierarquia, o que é fundamental quando se trabalha com sistemas de grande porte, além da representação de elementos básicos (reutilizáveis) com um comportamento conhecido utilizando objetos, teremos um *framework* adequado para simplificar e estruturar a representação e modelagem de sistemas automatizados.

Usando GHENeSys como ferramenta, associado a um método de modelagem e síntese das redes orientadas a objetos, podemos esperar uma redução no tempo de projeto e implementação, e na probabilidade de erros durante a etapa de projeto. No Capítulo 6 apresentaremos um estudo de caso onde poderemos mostrar estas vantagens.

6 ESTUDO DE CASO.

Para exemplificar o método de design proposto neste trabalho utilizando as redes GHENeSys escolhemos uma aplicação de automação predial. Esta aplicação foi escolhida por ser pequena o suficiente para mostrar o método proposto integralmente, em uma aplicação realistica, com certo grau de complexidade devido a integração, sem no entanto parecer um problema modelo simplificado. Neste caso fica também claro que a integração obtida é devido ao compartilhamento de informação e/ou recursos e não ao funcionamento conjunto dos subsistemas.

6.1 O problema e suas especificações.

A seguir apresentamos uma breve descrição tanto da parte física do sistema como da funcionalidade dos principais sub-sistemas.

Considere um pavimento formado por duas salas, nas quais deseja-se monitorar e controlar os sistemas de iluminação, segurança e conforto térmico. O equipamento instalado neste pavimento é mostrado na figura 6.1.

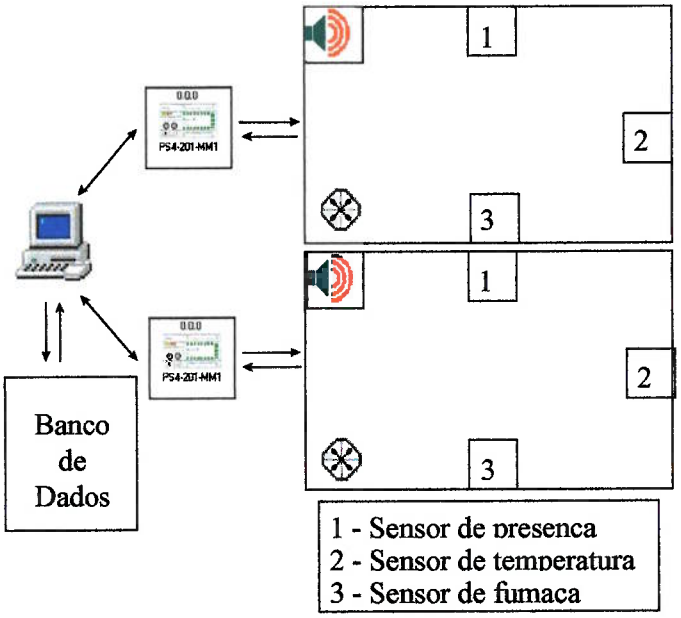


Fig 6.1 Esquema do andar

O comportamento desejado para cada um dos subsistemas será explicado a seguir.

6.1.1 Sistema de Segurança.

O sistema de segurança é responsável pela verificação de situações com presença não autorizada ou que possam afetar a segurança das pessoas e/ou instalações do ambiente em questão.

Na nossa experiência existem duas preocupações quanto à segurança: controle do acesso às salas e detecção de incêndio.

No tocante ao acesso às salas, o sistema permite que este seja habilitado ou não, conforme a política de utilização do local. A presença de uma pessoa num ambiente quando este

esteja desabilitado (no estado de presença não autorizada) é encarada como uma intrusão.

A alarme de incêndio ocorre quando é detectada a presença de fumaça e também a existência de uma fonte de calor. Esta última condição é detectada analisando as variações de temperatura nos últimos cinco minutos, assim como a resposta do controlador. A ocorrência de incêndio (eliminando-se o "alarme falso") está associada ao aumento sistemático da temperatura mesmo quando o sistema força a diminuição da temperatura padrão (set point).

O tratamento desta situação é feito nos níveis superiores da hierarquia de controle, já que nestes níveis é possível fazer uma análise mais abrangente, levando em conta por exemplo, a extensão da situação de incêndio ou o tipo da área afetada.

Para garantir um comportamento aceitável do sistema, os controladores dos níveis inferiores devem verificar se a comunicação entre o controlador na sala e o supervisório está danificada. Nesse caso o próprio controlador tomaria a decisão de ativar o alarme da sala em questão, antecipando assim uma possível falha no sistema de controle.

6.1.2 Sistema de iluminação.

A iluminação das salas é acionada quando houver presença de uma pessoa num ambiente que esteja habilitado (presença autorizada) e desligado em caso contrário.

6.1.3 Sistema de conforto térmico.

O sistema modelado é uma simplificação de um sistema real de conforto térmico em um ambiente predial, que envolve uma quantidade de instalações muito acima de uma única sala. Neste sistema cada sala possui um ventilador, que realiza o papel do conjunto "ventilador-*dampers*" num sistema real. A temperatura na sala é controlada variando a velocidade do ventilador, e assim o fluxo de ar resfriado.

O funcionamento do sistema de ventilação permite que o usuário defina uma temperatura (entre 20°C e 26°C). Sempre que houver presença permitida na sala o sistema é acionado de forma a manter a temperatura em uma faixa de ± 0.5 °C em torno da temperatura definida pelo usuário. Caso não haja ninguém na sala o sistema funcionará com velocidade constante e mais baixa, apenas para evitar um gasto excessivo de energia e uma situação de desconforto quando o sistema for acionado novamente.

O sistema de ventilação também deve enviar informações sobre o seu nível de funcionamento para o supervisório de nível superior, o que permite que o sistema de refrigeração funcione de forma mais eficiente. Por exemplo, no caso de várias salas estarem funcionando com seus "dampers" quase fechados (o qual corresponde a uma velocidade baixa do ventilador na nossa maquete) , pode-se atuar no fornecimento de ar frio a fim de diminuir o dispêndio de energia.

6.1.4 A arquitetura do sistema de controle.

A arquitetura de controle deste sistema é projetada utilizando uma arquitetura baseada em íntegrons. Esta arquitetura foi proposta por Ramos e Silva em [Ramos, 98]. Nesta aplicação cada subsistema será projetado como um íntegron que se encontra no nível mais baixo na nossa arquitetura. Estes íntegrons são agrupados observando as funções que devem cumprir individualmente e as informações que compartilhadas pelo conjunto para determinar que informações serão enviadas para os íntegrons de nível superior. Por exemplo, os subsistemas de segurança e conforto térmico compartilham os sinais provenientes dos sensores de presença e temperatura, e o subsistema de iluminação utiliza a sinal do sensor de presença que é propriedade do subsistema de segurança.

Entre as especificações para o funcionamento do sistema temos que:

- Cada nível de controle deve funcionar sem interrupções a não ser em caso de falha nos componentes.

- Em caso de falha em um dos subsistemas, o sistema deve continuar funcionando, ainda que com eficiência diminuída.

- O sistema deve ser flexível o suficiente, para permitir mudaças na estratégia de controle sem grandes modificações na arquitetura (flexibilidade).

A arquitetura do sistema de controle baseado em íntegrons é mostrado na figura 6.2.

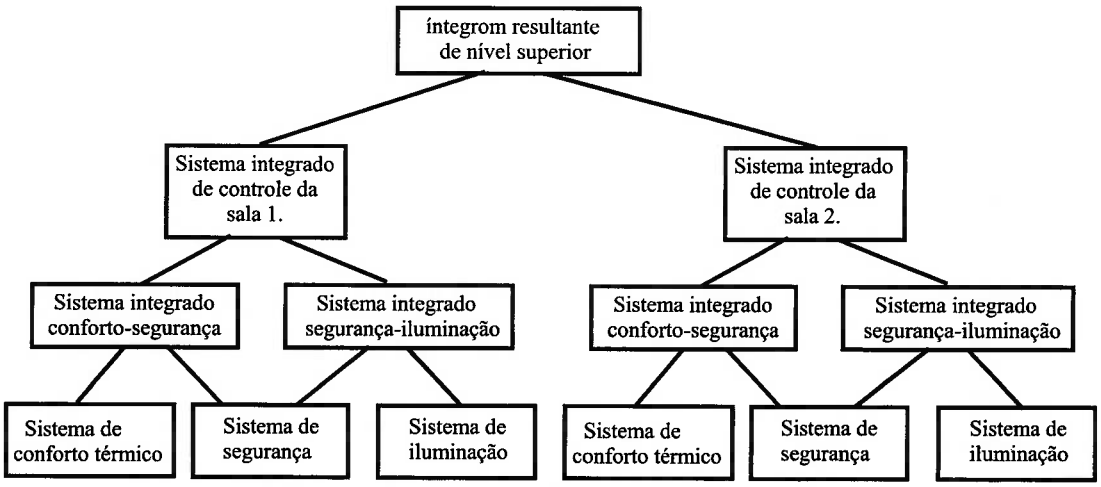


Fig. 6.2. Arquitetura baseada em íntegrons do sistema.

No experimento que resulta deste trabalho, o sistema supervisor rodará em um computador pessoal (PC) e os sistemas de controle de cada sala nos seus respetivos controladores programáveis.

6.2 O projeto do sistema.

Como na fase inicial de projeto ainda não temos definido qual equipamento executará uma determinada parte do programa de controle, a modelagem será feita atendendo a funcionalidade e integração em cada nível. No caso em que vários níveis de integração tenham de ser executados num mesmo equipamento só precisaremos mudar a rede no nível mais abstrato enquanto os módulos de controle poderão ser aproveitados.

Seguindo a orientação de DiCesare [DiCesare 93], um bom algorritmo de controle para sistemas de manufatura deve ser dividido em sua parte sequencial, geralmente chamada de planta, e na parte que embasa o sistema supervisório. Neste trabalho a modelagem será

dividido em dois níveis: supervisão (o controle do nível superior) e controle da sala.

6.2.1 O Modelo de dados do sistema.

O primeiro passo é definir a informação que será transferida entre os diferentes níveis e módulos. O objeto marca deve conter uma estrutura de dados (atributos) capaz de armazenar essa informação.

Na tabela 6.1 mostra-se a estrutura de dados do objeto *marca* que pertence a rede que representa o nível de controle de sala.

A estrutura do objeto *Marca* é a mesmo para as duas salas pois o funcionamento previsto também é o mesmo.

Nome do atributo	Observação
Presença	Indica o estado do sensor de presença da sala.
Fumaça	Indica o estado do sensor de fumaça da sala
Temperatura	Indica o valor da temperatura na sala
estado_sala	Indica o estado de habilitação da sala
Alarme_incêndio	Indica que um incêndio foi detectado na sala
estado_da_comunicação	Indica o estado da comunicação entre os níveis de controle de sala e integração de andar

Tabela 6.1 Estrutura do objeto *Marca*.

No controle de do nível superior, o objeto deve ter una estrutura capaz de armazenar os dados das duas salas. Como existe comunicação entre os níveis de controle, também definimos à estrutura do objeto *marca2* que representa a informação no nível superior. Os

objetos Sensor e Atuador são usados para representar as entradas/saídas com o processo.

Na tabela 6.2 mostra-se a estrutura de dados do objeto *Marca2*

Nome do atributo	Observação
Dados sala 1	Apontador para a copia do objeto *marca* da sala 1.
Dados sala 2	Apontador para a copia do objeto *marca* da sala 2.
Comando1	Indica o estado desejado para a sala 1
Comando2	Indica o estado desejado para a sala 2
Incêndio1	Indica que um incêndio foi detectado na sala 1
Incêndio2	Indica que um incêndio foi detectado na sala 2
Estado_da_comunicação1	Indica o estado da comunicação entre o nivel superior e a sala 1
Estado_da_comunicação2	Indica o estado da comunicação entre o nivel superior e a sala 2

Tabela 6.2 Estrutura do objeto *marca2*.

6.2.2 O controle das salas.

Neste nível de controle o sistema deve estar preparado para ler as sinais dos sensores que são uma indicação do estado das salas e de manipular as sinais dos atuadores para garantir que as salas se comportem segundo a política de uso adotada. Como foi mostrado no epígrafe anterior, existe uma troca de informações entre os níveis de controle, seguindo a estrutura proposta por Ramadge e Wonham [Ramadge, 87] para controle supervisorio, e que é a base dos íntegrons definidos por Ramos e Silva [Ramos, 98].

A modelagem do controle mais a planta do íntegron do controle de sala é mostrado na figura 6.3.

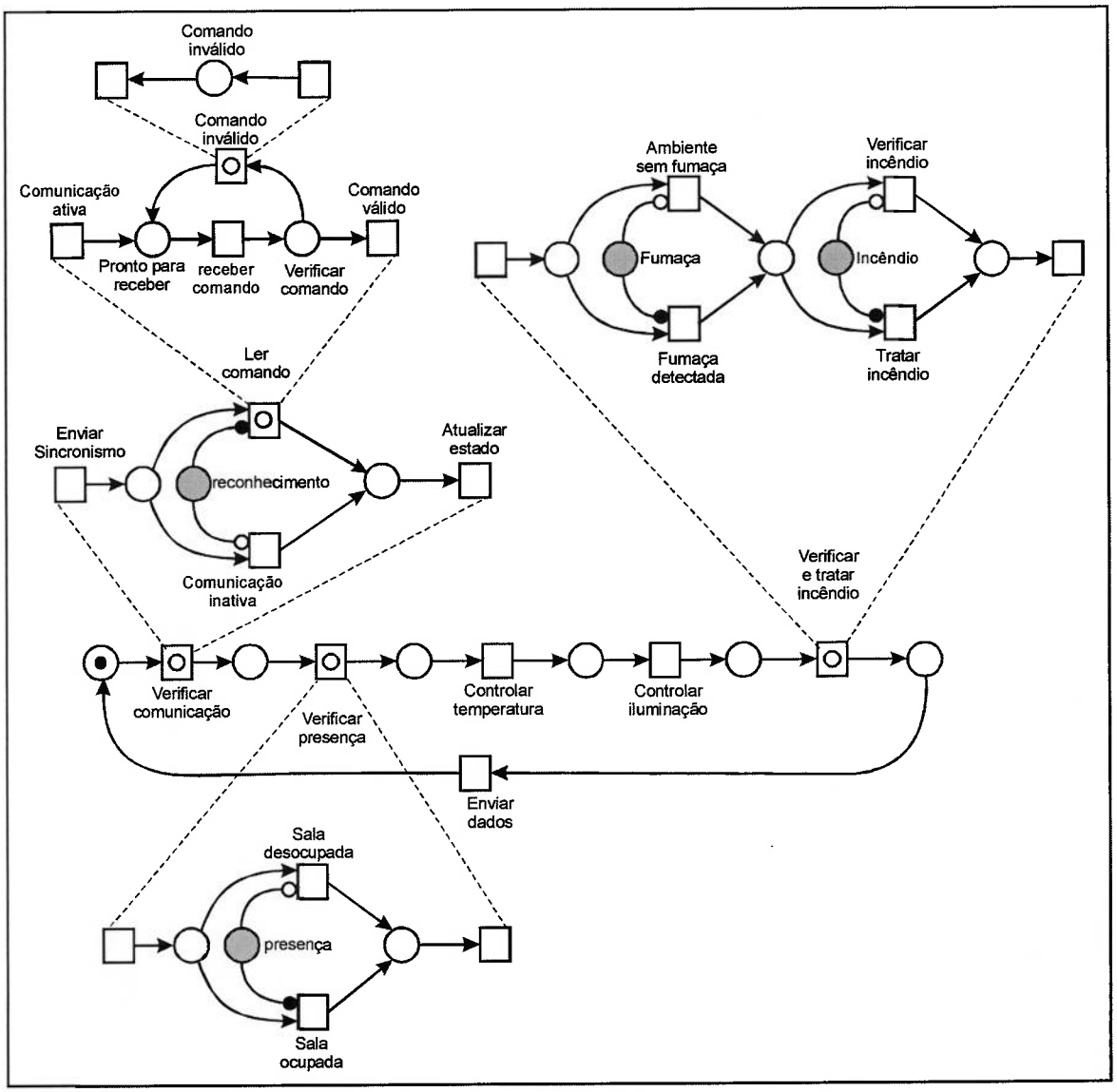


Fig 6.3 Modelo do integron do controle da sala.

Note-se que as redes e subredes utilizadas na modelagem do controle da sala são todas redes vivas, portanto o sistema é livre de bloqueios. Este modelo fornece a ordem em que os métodos de adquisição de dados, controle e transferência de informação devem ser executados.

Uma vez verificado que o modelo têm o comportamento especificado para o sistema, um refinamento mais detalhado é necessário para aproximá-lo da fase de implementação. Este refinamento é colocado nos métodos de cada objeto.

Na tabela 6.3 aparece uma breve descrição dos métodos dos objetos da figura 6.3.

Inicialização dos atributos	**Breve descrição do método ExecutaTask(Marca)**
Name: Comunicação_inativa *NetName*:Verificar_Comunicação *EstimatedTime*: 0.5	Marca.estado_da_comunicação=inativa
Name: Comunicação_ativa *NetName*:Ler_Comando *EstimatedTime*: 0.5	Marca.estado_da_comunicação=ativa
Name: receber_comando *NetName*:Ler_Comando *EstimatedTime*: 0	Se Marca.estado_da_comunicação=ativa Então Enviar_comando.getToken(Marca2) Comando=Marca2.comando1
Name: Verificar_comando *NetName*:Ler_Comando *EstimatedTime*: 0 *Capacity*: 1 *MarkNumber*: 0	Comando_valido.EstimatedTime=0 Comando_invalido.EstimatedTime=0 Se Marca.estado_da_comunicação=ativa Então Se Tentativa < 3 Então Se IsValid(Comando) Então Comando_invalido.EstimatedTime=1 Marca.estado_sala=Extract(Comando) Init(Comando) Tentativa = 0 Senão Tentativa = Tentativa + 1 Marca.estado_sala = null Comando_valido.EstimatedTime=1 Senão Marca.estado_da_comunicação=inativa Tentativa = 0 Senão Comando_invalido.EstimatedTime=1
Name: Sala desocupada *NetName*: Verificar_presença *EstimatedTime*: 0	Marca.Presença= Sensor.Presença Se Marca.estado_sala <> habilitada Então Atuador.Alarme_Intruso = 1
Name: Sala ocupada *NetName*: Verificar_presença *EstimatedTime*: 0.1	Marca.Presença= Sensor.Presença Se Marca.estado_sala = habilitada Então Atuador.Alarme_Intruso = 0

Tabela 6.3 Breve descripção dos métodos usados no controle da sala

Inicialização dos atributos	Breve descripção do método ExecutaTask(Marca)
Name: Controlar temperatura *NetName*: Sala1 *EstimatedTime*: 0.1	Se Marca.estado_sala = habilitada Então Marca.Temperatura= Sensor.temperatura SetPointO=ExtractSP(Comando) SetPointD=ExtractSP(Comando) Se (Marca.presença = 1) Então Se Marca.Temperatura > SetPointO Então Atuador.VelVent = Atuador.VelVent + 1 Senão Atuador.VelVent = Atuador.VelVent – 1 Senão Se Marca.Temperatura > SetPointO Então Atuador.VelVent = Atuador.VelVent + 1 Senão Atuador.VelVent = Atuador.VelVent – 1 Senão Atuador.VelVent = 0
Name: Ambiente sem fumaça *NetName*:Verificar e tratar incênd. *EstimatedTime*: 0	Marca.Fumaça= Sensor.Fumaça
Name: Fumaça detectada *NetName*:Verificar e tratar incênd. *EstimatedTime*: 0	Marca.Fumaça= Sensor.Fumaça
Name: Controlar iluminação *NetName*: Sala1 *EstimatedTime*: 0.1	Se (Marca.Presença = 1) e (Marca.estado_sala = habilitada) e (Horario = Nocturno) Então Atuador.Luz = 1 Senão Atuador.Luz = 0
Name: Verificar incêndio *NetName*:Verificar e tratar incênd. *EstimatedTime*: 0	Se Marca.Fumaça= 1 Então Se ExistHeatFocus() Então Atuador.VelVent = 0 Atuador.Duchas = 1 Atuador.RotaFuga = 1 Atuador.Luz = 0
Name: Tratar incêndio *NetName*:Verificar e tratar incênd. *EstimatedTime*: 0	Atuador.VelVent = 0 Atuador.Duchas = 1 Atuador.RotaFuga = 1 Atuador.Luz = 0

Tabela 6.3 Breve descripção dos métodos usados no controle da sala (Continuação)

6.2.3 O controle do nível superior.

No nível superior de controle não se têm accesso de forma direta aos objetos sensores e

atuadores, pois estes são elementos presentes na sala e portanto accessíveis somente pelo controlador associado a esta. Desta forma, a execução da política de controle determinada para as salas, será elaborada baseada na informação fornecida pelo nível de controle de sala e os comandos do operador do sistema.

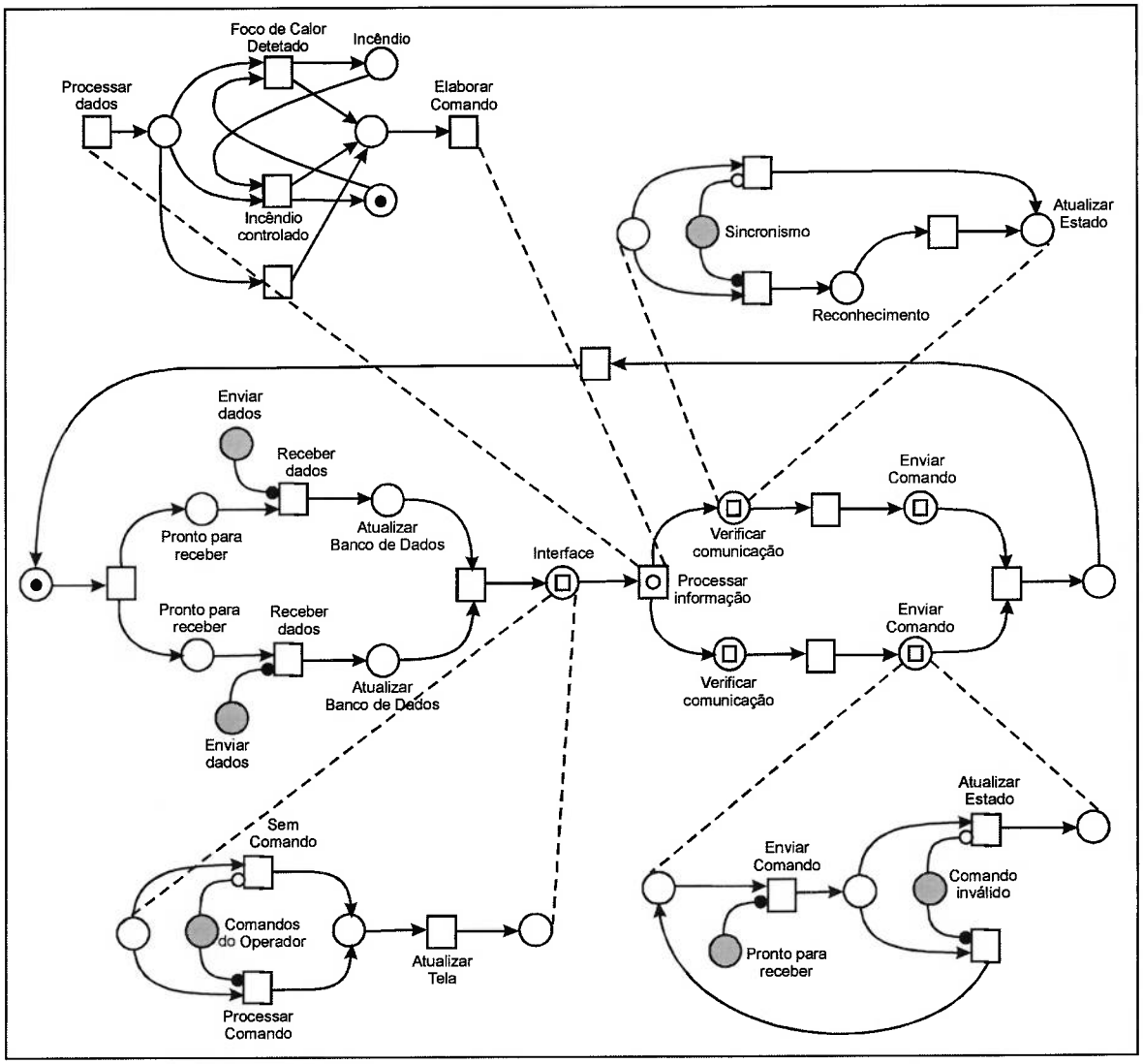


Fig. 6.4 Modelo do controle de nível superior.

Adotando o mesmo procedimento usado na modelagem do controle de sala construímos o

modelo que reflete as funcionalidades do nível superior. Este modelo é mostrado na figura 6.4.

O objeto *Marca2* apresentado anteriormente é o objeto base deste nível de controle e, como vimos anteriormente é usado também para passar os comandos ao nível de controle da sala.

A seguir, apresentamos na tabela 6.4 uma breve descrição dos métodos utilizados na modelagem do controle de nível superior.

Inicialização dos atributos	**Breve descripção do método ExecutaTask(Marca2)**
Name: Receber dados *NetName*: Supervisório *EstimatedTime*: 0.1	Se (Marca2.Estado_da_comunicação1 é ativa) Então EnviarDados.GetToken(Marca) Marca2.DadosSala1←Marca
Name:Atualizar Banco de Dados *NetName*: Supervisório *EstimatedTime*: 0.1 Capacity: 1 MarkNumber: 0	Se (Marca2.DadosSala1<> null) Então Inserir no BD dados Marca2.DadosSala1
Name: Processar Comando *NetName*:Interface *EstimatedTime*: 0	Comandos do Operador.GetToken(ManipEventos) Mientras ManipEvent.NumEventos > 0 fazer Marca2.Comando1=AtivarComando(ManipEventos.Evento)
Name: SemComando *NetName*:Interface *EstimatedTime*: 0	Marca2.Comando1= AtivarComando(nada)
Name: Processar Dados *NetName*:Processar Informação *EstimatedTime*: 0	FocoCalor = False Se (Marca2.DadosSala1.fumaça= 1) Então VerificarFocoCalor(FocoCalor)
Name: Foco de Calor Detetado *NetName*:Processar Informação *EstimatedTime*: 0	Marca2.Incêndio1 = 1
Name: Incêndio controlado *NetName*:Processar Informação *EstimatedTime*: 0	Marca2.Incêndio1 = 0
Name: Elaborar Comando *NetName*:Processar Informação *EstimatedTime*: 0	Se (Marca2.Incêndio1 = 1) Então Marca2.Comando1=AtivarComando(apagar incêndio) Senão Marca2.Comando1=AtivarComando(normal)

Tabela 6.4 Descrição dos métodos do modelo do nível superior de controle.

Neste modelo, a maioria das estruturas utilizadas, resultam da composição das estruturas básicas do primeiro grupo citados no capítulo anterior. A subrede que representa o elemento macro **Processar Informação** é a única subrede que não foi originada da combinação das estruturas próprias e portanto requer de uma análise mais aprofundada.

Para aprofundar na análise desta subrede incluiremos um elemento box conectado entre as atividades de entrada e saída, tornando-a uma rede cíclica. (Ver figura 6.5)

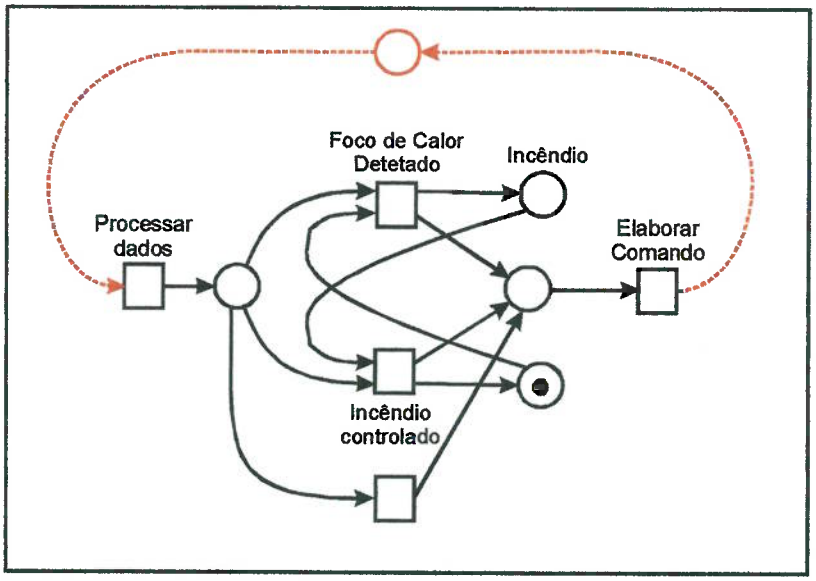


Fig. 6.5 Subrede do elemento macro **Processar informação** em sua versão cíclica.

A matriz de incidência desta subrede modificada seria:

$$C = \begin{vmatrix} -1 & 1 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & -1 & -1 & -1 & 1 \\ 0 & 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \end{vmatrix}$$

Calculando as invariantes de atividade obtém-se os seguintes resultados:

$$x = \begin{vmatrix} 0 & 1 & 2 & 3... \\ 0 & 0 & 1 & 1... \\ 0 & 0 & 1 & 1... \\ 0 & 1 & 0 & 1... \\ 0 & 1 & 2 & 3... \end{vmatrix}$$

Esta solução mostra que, sempre que disparada a atividade a_1, a atividade a_5 também será disparada o que mostra que esta rede também é viva e portanto é também uma rede própria (ainda que não seja produto da composição de redes próprias).

Com esta verificação, garantimos que todas as subredes e a rede no nível mais abstrato são vivas, portanto a rede resultante dos refinamentos também é viva.

6.3 Resultados.

A rede GHENeSys foi utilizada para modelar um sistema modelo de automação predial, onde se tenta integrar as informações dos módulos de segurança, incêndio, HVAC e iluminação. Os sistemas compartilham informação (sobre presença de pessoas por exemplo) de modo a evitar a colocação de sensores de forma redundante.

Comparando a modelagem feita neste trabalho com a modelagem apresentada em [González, 99] do mesmo sistema, podemos observar uma diminuição do número de elementos utilizados na modelagem, apesar de conter maiores informações que facilitam a

implementação do sistema de controle, produto do encapsulado de informação nos métodos.

Devido a técnica de modelagem utilizada, fica mais fácil identificar onde devemos colocar mecanismos de controle mais estritos para conservar o fluxo de controle do sistema. Por exemplo, no refinamento do elemento **Ler comando**, (Ver figura 6.3) existe a possibilidade (apesar de ser uma rede própria) de ficar num laço e o fluxo de controle ficar interrompido. Neste caso, o método ExecuteTask do elemento **Verificar comando** possue um mecanismo adicional que permite controlar o número de vezes que o laço é executado e garante a saída desta subrede mesmo que o comando não possa ser lido por causa de uma falha na comunicação entre os níveis de controle.

A integração entre os diferentes níveis de controle também é abordada de forma explícita pelo modelo GHENeSys, pois a troca de informações entre estes níveis aparece na forma de relação de fluxo direta ou através do pseudoboxes, e permite, já na fase de simulação, detectar possíveis falhas de comunicação no sistema.

Como uma das principais fontes de falhas nos sistemas de controle é justamente a presença de bloqueios, seguindo a disciplina de modelagem proposta, este tipo de falhas é eliminado no processo de construção do modelo, ficando somente aquelas que resultam do compartilhamento de dados que estão claramente identificados (pela presença dos *pseudoboxes*) e podem ser corrigidos ou aprimorados ainda na etapa de modelagem/simulação.

7 CONCLUSÕES E TRABALHO FUTURO.

7.1 Conclusões.

Neste trabalho foi apresentada uma rede estendida orientada a objetos para o *design* de sistemas discretos. Esta rede foi testada para modelar um sistema de controle predial onde a integração é necessaria para evitar a colocação redundante de sensores – além de admitir o uso flexível de políticas de uso das dependências - e também foi utilizada para modelar exemplos de workflow encontrados na literatura, notadamente no trabalho de [Van der Aalst, 97].

A experiência no uso desta rede, mesmo sendo restrita até o momento, mostra que há de fato uma diferença marcante entre redes com um formalismo *sound,* com algorítmos bem definidos de cálculo de propriedades – como a rede colorida – e redes estendidas baseadas em redes elementares. As primeiras facilitam o cálculo de propriedades (invariantes) e a abstração, no entanto a interpretação dos resultados é bastante difícil. No segundo caso a modelagem é bastante difícil (por falta de abstração e métodos de síntese), mas a interpretação é fácil e direta, embora fique mais e mais comprometida na medida em que o sistema cresce. Para este tipo de rede, uma modularização prolonga sua eficácia tornando-

as atraentes para algumas aplicações – principalmente nos casos em que a própria modularização é direcionada a este tipo de aplicação.

Tudo indica que, o ideal seria uma posição intermediária que preservasse a interpretação e o uso da intuição no processo de modelagem e análise (principalmente para sistemas inteiramente novos); e por outro lado, que permitisse a abstração e o dobramento das redes de alto nível, preservando a possibilidade do cálculo de propriedades (invariantes, distância síncrona, vivacidade, etc.).

Além disso a abordagem orientada a objetos, além de ter uma estrutura hierárquica disciplinada, traz também a vantagem da reutilização de modelos *bottom up* (ao tempo em que a modelagem conceitual segue por refinamentos). Portanto é possível obter uma fusão de métodos *top down* e *bottom-up* utilizando um esquema orientado a objetos.

O que tentamos fazer com a rede GHENeSys é justamente buscar um equacionamento entre a preservação da abordagem direta e intuitiva em um esquema de grafo bipartido (rede) e a sofisticação das redes de alto nível (seja utilizando lógica, teoria de tipos ou objetos). Outro objetivo importante é manter a capacidade de analisar diversos sistemas de controle (presentes em sistemas de manufatura) como o controle centralizado, hierárquico, hierárquico modificado e distriuido ou multi-agente, e ainda facilitar o processo de síntese das redes.

Além de permitir a análise estes diferentes sistemas de controle, a GHENeSys, mediante a abordagem orientada a objetos, facilita a passagem do modelo para a implementação,

diminuindo assim o tempo de desenvolvimento do sistema.

A disciplina de construção dos modelos, unido a prova da conservação de certas propriedades durante o processo de refinamento, facilita de forma consideravél a validação dos modelos, o que pode ser feito em qualquer nível de abstração.

7.2 Trabalhos futuros.

Neste trabalho foram apresentadas as bases teóricas para a construção de um editor-simulador das redes GHENeSys. O primeiro passo á partir deste ponto é a implementação deste editor-simulador, que por sua vez seria o embrião de uma ferramenta de análise de sistemas integrados e flexíveis em geral, podendo ser aplicada a sistemas de manufatura ou a outros sistemas com as mesmas características.

No exposto neste trabalho, o objeto marca foi considerado de "tipo único" para que a regra de disparo da rede não fosse modificada. Em trabalhos futuros, tentaremos converter o objeto marca em uma classe, onde os valores dos atributos podem se modificar (através de métodos) durante a execução da rede. Desta forma estaremos partindo para um tipo de redes coloridas, só que utilizando a orientação a objetos, em vez da teoria de tipos para sustentar o formalismo da rede.

Existe na literatura algumas propostas de formalismo para objetos, porém nenhum deles é unanimidade no mundo académico. Apesar disso existe um em particular proposto por Hubbers e Hofstede usando álgebra de processos, que concentra vários pontos em comum

com os demais, e que usamos como referência para a nossa proposta de Redes de Petri por objetos.

Ao lado disto, pretendemos estudar e desenvolver métodos algébricos para a análise de propriedades que sustentem a utilização desta rede para design de sistemas complexos. Este é também um dos motivos para escolher um formalismo algébrico para objetos e uma contricuição que pretendemos dar no sentido de entender a fomentar uma efetiva unidade entre as redes estendidas e de alto nível.

8 REFERÊNCIAS BIBLIOGRÁFICAS.

[Agha, 86] Agha, G. **A Model of Concurrent Computation in Distributed Systems**. *MIT Press*, Cambridge, 1986.

[Bastide, 95] Bastide R. **Approaches in unifying Petri nets and the Object-Oriented Approach.** *Proc. 1st Workshop on Object-Oriented Programming and Models of Concurrency*, June 1995.

[Battiston, 90] Battiston E, De Cindio F, Mauri G. **OBJSA Nets: a class of high level nets having objects as domain.** *Lecture Notes in Computer Science 340*, Springer-Verlag, 1990.

[Bernardinello, 92] Bernardinello L, De Cindio F. **A Survey of Basic Models and Modular Net Classes.** *Lecture Notes in Computer Science 609*, Springer-Verlag, 1992.

[Best, 87] Best E. **Struture Theory of Petri Nets: The Free Choice Hiatus.** *Lecture Notes in Computer Science 254*, Springer-Verlag, 1987.

[Best, 90] Best E. **Design Methods Based on Nets.** *Lecture Notes in Computer Science 424*, Springer-Verlag, 1990.

[Bibertein, 95] Biberstein O., Buchs D. **Structured Algebraic Nets with Object-Orientation**. *Proc. 1st Workshop on Object-Oriented Programming and Models of Concurrency*, June 1995.

[Booch, 90] Booch G. **Object Oriented Annalysis and Design with Applications**, *The Benjamin/Cummings Publishing Company Inc.* RedWood City, California, 1991.

[Brauer, 90] Brauer W., Gold R., Vogler W. **A Survey of Behaviour and Equivalence Preserving Refinements of Petri Nets.** *Lecture Notes in Computer Science 424*, Springer-Verlag, 1990.

[Cao, 90] Cao, Xi-Ren; Ho, Yu-Chi. **Models of Discrete Event Dynamic Systems.** *IEEE Control Systems Magazine*, 1990.

[Colom, 90] Colom, J.M.; Chiola, G.; Silva, M. **On liveness Analysis Through Linear Algebraic Techniques.** Departamento de Ingenieria Electrica e Informática, Universidad de Zaragoza, Research Report GISI-RR-90-11, 1990.

[Desel, 92] Desel J. **A Proof of the Rank Theorem for Extended Free Choice Nets**. *Lecture Notes in Computer Science 616*, Springer-Verlag, 1992.

[Desel & Esparza, 95] Desel J, Esparza J. **Free-Choice Petri Nets**. *Vol 40 of Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press, Cambridge, 1995.

[DiCesare, 93] DiCesare, T., et al, **Practice of Petri Nets in Manufacturing**, Chapman & Hall, 1993.

[Dilts, 91] Dilts, D., Boyd, N.P., Whorms, H.H. **The evolution of Control Architecture Automated Manufacturing Systems**, *Journal of Manufacturing Systems*, 10, 79-93, 1991.

[Esparza & Silva, 90] Esparza J. Silva, M. **Circuits, Handles, Bridges and Nets.** *Lecture Notes in Computer Science 483*, Springer-Verlag, 1990.

[Esparza, 90] Esparza J. Silva, M. **On the Analysis and Synthesis of Free Choice Systems.** *Lecture Notes in Computer Science 483*, Springer-Verlag, 1990.

[Esparza, 92] Esparza J. Silva, M. **A Polinomial-Time Algorithm to Decide Liveness of Bounded Free Choice Nets.** *Theoretical Computer Science*, 1992.

[Esparza, 00] Esparza J. Melzer E. **Verification of safety properties using integer programming: Beyond the state equation**. *Formal Methods in System Design*, 16:159-189, 2000.

[Esser, 97] Esser R. **An Object-Oriented Petri Net Language for Embedded System Design.** *Proc. of the 8th International Workshop on Software Engineering Practice (STEP)*, London, England, July 1997.

[Fehling, 93] Fehling R. A **Concept of Hierarchical Petri Nets with Buildings Blocks** *Lecture Notes in Computer Science 674*, 1993.

[Genrich, 83] Genrich, H., Lautenbach, K; **S-Invariance in Predicate/Transition Nets**, in *Aplication and Teory of Petri Nets*, A. Pagnoni and G. Rozemberg (eds.), Springer Pub. Co, 1983.

[Goltz, 86] Goltz, U; Chong-Yi, Y. **Synchronic Structure**, *Lecture Notes in Computer Science*. Springer-Verlag, pp.232-252, 1986.

[González, 99] González P. M., Silva J. R. **Sistema Supervisório Desentralizado Basado en Íntegrons** Memórias del III Workshop sobre Sistemas inteligentes para edificios. (SIntEd-CYTED). Cancún, Mexico. pp.95-106, 1999. (en español).

[Hasegawa, 87] Hasegawa, K. et al. **Simulation of Discrete Production Systems Based on Mark Flow Graph.** *System Science*, Vol 13, Poland, 1987.

[Hasegawa, 96] Hasegawa, K. **Modeling, Control and Deadlock Avoidance of Flexible Manufacturing Systems.** *Conferencias Plenárias of XI CBA*, SBA, pp 37-51, 1996.

[Hubbers & Hofstede, 97] Hubbers J.W.G.M., Hofstede A.H.M. ter. **Formalization of Communication and Behaviour in Object-Oriented Analysis**. *Data & Knowledge Engineering*, 23(2):147--184, August 1997.

[Jacobson, 92] Jacobson I, Christerson M, Jonsson M, and van Overgaard P. **OO Software Engineering, A Use Case Driven Approach.** Addison-Wesley, Reading, Massachusets, 1992.

[Jensen, 90] Jensen, K. **Colored Petri nets: a high level language for system design and analysis**. *Lecture Notes in Computer Science 483*, Springer-Verlag, 1990.

[Jensen, 96] Jensen, K. **Colored Petri Nets: Basic Concepts, Analysis Methods and Practical Use**. Volume 1, Second Edition, Springer-Verlag, 1996.

[Lakos, 94] Lakos C. A, Keen C.D. **LOOPN++: A new language for Object Oriented Petri Nets**, Proceedings of Modeling and Simulation (European Simulation Multiconference), pp369-374, Barcelona, 1994.

[Lakos, 95] Lakos C. A. **The Object Orientation of Object Petri Nets.** Deparment of Computer Sciences, University of Tasmania, 1995.

[Lin&Wonhan, 90] Lin, F. and Wonham, W.M.; **Decentralized Control and Coordination of Discrete-event with Practical Observation**, *IEEE Transactions on Automatic Control*, vol. 35, no. 12, 1330-1337, 1990.

[Linger, 79] Linger R. C., Mills H. D., Witt B. I. **Structured Programming**, Addison-Wesley Publishing Company, 1979.

[Martinez, 86] Martinez J., Alla H., Silva M. **Petri Nets for the specification of Flexible Manufacturing Systems**. *in Modeling and Design of Flexible Manufacturing Systems*. New York: Elsevier Science Publications, pp.389-406, 1986.

[Miyagi, 88] Miyagi P. E. **Control System Design, Analysis and Implementation of Discrete Event Production Systems by Using Mark Flow Graph**. PhD Thesis, Tokyo Institute of Technology, Tokyo, 1988.

[Moldt, 95] Moldt D. **OOA and Petri Nets for System Specification.** *Proceedings of 16th International Conference on Application and Theory of Petri Nets, Lecture Notes in Computer Science.* Springer-Verlag. Turin, Italy, 26.-30. Jun, 1995

[Murata, 89] Murata T. **Petri nets: properties, analysis, and applications.** *Proc. IEEE* pp 541-580. 1989.

[Oxford 86] Oxford University Press. **Dictionay of Computing.** *Oxford University Press*, 1986.

[Petri, 73] Petri C. A. **Concepts of Net Theory**. *Mathematical Foundations of Computer Science.* Proceedings of Symposium and Summer School, High Tatras, 1973.

[Prock, 91] Prock J. **A New Technique for Fault Detection Using Petri Nets.** *Automática, Vol 27 No 2*, pp. 239-245, 1991.

[Ramadge, 87] Ramadge, P. J.; Wonham, W. M. **Supervisory Control of a Class of Discrete Event Processes**. *SIAM Journal of Control Optimization. Vol 25 No 1,* pp 206-230. 1987.

[Ramos, 98] Ramos, R.L.C.B. Silva, J.R. **A Formal Model for Integrated Complex Dynamic Systems.** 5th IFAC *Workshop on Intelligent Manufacturing Systems* - IMS'98. Gramado/Canela – RS, Brazil. November 9-11, 1998.

[Reisig, 82] Reisig W. **Petri Nets An Introduction**, Springer-Verlag, 1982.

[Rumbaugh at al, 91] Rumbaugh J, Blaha M, Premerlani W, Eddy F, and Lorenson W. **Object-Oriented Modeling and Design**. *Prentice-Hall*, New Jersey, 1991.

[Santos Filho, 93] Santos Filho, D. J. **Proposta de Mark Flow Graph Estendido para a Modelagem e Controle de Sistemas Integrados de Manufatura.** Disertação (Mestrado), Escola Politécnica, USP, 1993.

[Sibertin-Blanc, 85] Sibertin-Blanc C. **High level Petri Nets with Data Structure.** *6th European workshop on Petri Nets an Applications,* Espoo, Finland, June 1985.

[Sibertin-Blanc, 98] Sibertin-Blanc C. **CoOperative Objects: Principles, Use and Implementation** *Lecture Notes in Computer Science XX*, Springer-Verlag, 1998.

[Singh, 98] Singh, N. **Systems Approach to Computer-Integrated Design and Manufacturing**. John Wiley & Sons, Inc, 1998.

[Silva, 85] Silva M. **Las Redes de Petri en la Automática y la Informática.** *Editorial AC*, Madrid 1985.

[Silva, 92] Silva J. R. Pessoa F. J. B. **Análise Semi-automática de Mark Flow Graph**, *Ibero-Americam Workshop in Autonomous Systems Robotics and CIM*, Lisbon, 1992.

[Silva & Miyagi, 95] Silva J. R., Miyagi P. E. **PFS/MFG: A High Level Net for the Modeling of Discrete Manufacturing Systems**. *Balanced Automation Systems, Arquitectures and Design Methods* Chapman & Hall, Eds Luis Camarinha-Matos and Hamideh Afsarmanesh pp349-362.1995.

[Silva, 96] Silva, J.R., Miyagi, P.E. **A formal approach to PFS/MFG: a Petri net representation of discrete manufacturing systems**, *Studies in Informatics and Control*, IC Publications, Romania. 1996.

[Silva, 98] Silva J.R. **Interactive Design of Integrated Systems**, *Intelligent Systems for Manufacturing*, L.M.Camarinha-Matos, H. Afsarmanesh, V. Marik, (eds.), Kluwer Academic Pub. 1998.

[Suzuki, 83] Suzuki I., Murata T. **A Method for stepwise refinement and abstraction of Petri Nets.** *Journal of Computer Systems Science* Vol 27, 1983.

[Thiagarajan, 87] Thiagarajan P. S. **Elementary Nets Systems.** *Lecture Notes in Computer Science 254*, Springer-Verlag, 1987.

[Valette, 79] Valette R. **Analysis of Petri Nets by Stepwise Refinement.** *Journal of Computer Systems Science* Vol 18, 1979.

[Valette, 90] Valette R; Silva M. **A Rede de Petri : Uma Ferramenta para a Automação Fabril.** In: *4.o Congresso Nacional de Automação Industrial*. Anais. São Paulo, 1990.

[van der Aalst, 97] van der Aalst W. M. P. **Verification of Workflow Nets.** *Lecture Notes in Computer Science 1248*, Springer-Verlag. 1997.

[van der Aalst, 98] van der Aalst W. M. P. **The Application of Petri Nets to Workflow Management.** *Journal of Circuits, Systems, and Computers, Vol 8, No 1.* 1998.

[Vogler, 87] Vogler W. **Behavior preserving refinements of Petri Nets.** *Lecture Notes in Computer Science 246*, Springer-Verlag. 1987.

[Webster's, 77] Webster's. **Webster's New Twentieth Century Dictionary,** *Collins World*, 1977.

[WFMC, 96] WFMC, **Workflow Management Coalition Terminology and Glossary** *(WFMC-TC-1011), Technical report, Workflow Management Coalition*, Brussels, 1996.

Apêndice I – Programas em MATLAB utilizados para calcular marcações e determinar vetor de habilitação.

```
function M1 = getnextmarking(C,M,D,t)
%
% Calcula a nova marcação da rede
%
% entradas: C - matriz de incidencia
%           M - marcação atual
%           D - matriz diagonal modificada
%           t - vetor de disparo
% saída:   M1 - nova marcação
%
%   GHENeSys Net                              Author:PGF
%   Created: 28-03-2001                       Revised:
%
M1 = M + D*C*t;
[l,a]=size(M1);
for i=1:a
   for j=1:l
      if (M1(j,i) < 0)
         disp('Tem eventos em conflito');
         M1 = M;
         return
      end
   end
end


function D = getmatrizD(p)
%
% getmatrizD calcula a matriz D levando em consideração o
numero
% de boxes (b) e pseudoboxes(p) na rede
% entrada: p - vector que identifica os pseudoboxes
%   exemplo: [1,1,1,1,1,0,0] para os elementos
b1,b2,b3,b4,b5,p1,p2
% saída:    D - matriz D
%
%   GHENeSys Net                              Author:PGF
%   Created: 28-03-2001                       Revised:
%
[m,n]=size(p);
if m ~= 1
   disp('Erro no vetor de lugares');
else
   D=diag(p);
end
```

Apêndice I – Programas em MATLAB utilizados para calcular marcações e determinar vetor de habilitação. (Continuação)

```
function y = isenabled(C,M,K)
%
% isenabled calcula o vetor de habilitação da rede
%
% entradas: C - matriz de incidencia
%           M - marcação atual
%           K - capacidade dos lugares
% saída:    y - vetor de marcação
%
%   GHENeSys Net                              Author:PGF
%   Created: 28-03-2001                       Revised:
%
[l,a]=size(C);
B = ones(1,a);
S = M*B + C;
y = ones(a,1)
for i=1:a
   for j=1:l
      if (S(j,i) < 0) | (S(j,i) > K(j,1))
         y(i)=0;
      end
   end
end
```