ROSSINI SÁLVIO BOMFIM DOS SANTOS

Modelagem e Análise de Performance de Sistemas Flexíveis de Manufatura Baseado em Redes de Petri Temporizadas: Estudo de Caso na Indústria Automobilística

São Paulo

2008

Rossini Sálvio Bomfim dos Santos

Modelagem e Análise de Performance de Sistemas Flexíveis de Manufatura Baseado em Redes de Petri Temporizadas: Estudo de Caso na Indústria Automobilística

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

São Paulo
2008

Rossini Sálvio Bomfim dos Santos

Modelagem e Análise de Performance de Sistemas Flexíveis de Manufatura Baseado em Redes de Petri Temporizadas: Estudo de Caso na Indústria Automobilística

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

Programa: Eng. Mecânica

Área de Concentração:
Controle e Automação

Orientador:
Prof. Dr. José Reinaldo Silva

São Paulo
2008

FICHA CATALOGRÁFICA

Santos, Rossini Sálvio Bomfim dos
Modelagem e análise de performance de sistemas flexíveis de manufatura baseado em redes de petri temporizadas : estudo de caso na indústria automobilística / R.S.B. dos Santos. -- São Paulo, 2008.
115 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.Redes de petri 2.Modelagem matemática 3.Simulação 4.Análise de desempenho I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t.

AGRADECIMENTOS

À Deus pela Benção de Viver.

Aos meus pais por todos os valores ensinados e preparação para a vida.

Aos meus irmãos por todo carinho e incentivo.

À minha esposa pela paciência, incentivo, dedicação e compartilhamento da vida.

À minha filha, Ana Beatriz, por toda alegria, entusiasmo, e entendimento do que é a vida e, do que é viver, colocando no meu horizonte uma estrela que me guia ao longo da jornada para a conquista de meus objetivos e realização dos meus deveres.

Ao meu orientador, pela amizade, paciência e dedicação conjunta para a finalização deste trabalho.

Aos meus amigos pela compreensão e encorajamento.

Ninguém que deseje a verdade, por mais errado que seja o caminho escolhido para sua busca, será abandonado.

(Krishna)

RESUMO

A necessidade de aumento de produção, da redução de custos e do aumento da qualidade de bens de consumo, tem motivado a constante evolução dos sistemas de produção, migrando os tradicionais sistemas de produção para os modernos e complexos sistemas de manufatura, onde a performance depende da eficiência dos equipamentos e do controle do processo. Por outro lado, a eficiência dos equipamentos depende de sua confiabilidade e manutenabilidade. Neste trabalho a análise de performance é avaliada com o uso de Rede de Petri p-t-Temporizada e através de simulações, incluindo a avaliação da confiabilidade do processo pela análise da otimização da saída do sistema, isto é, quantidade de itens produzidos. Nesta abordagem, uma lógica linear foi desenvolvida e validada utilizando-se uma comparação de resultados das classes de estados do algoritmo proposto com a ferramenta de simulação Tina para um modelo de um esquema produtor – consumidor. Apresenta-se um estudo de caso na indústria automotiva, consistindo na análise dos problemas reais enfrentados em uma fábrica de carrocerias, com o uso da Rede de Petri p-t-Temporizada.

Palavras-chave: Rede de Petri Temporizada. Sistema Flexível de Manufatura. Modelagem. Simulação. Análise de Performance.

ABSTRACT

The necessity of growing in production, with reduction of costs and improvement in the quality of consumption good, has motivated the constant evolution of production systems, transforming traditional production systems into the modern and complex manufacturing systems, where the performance depends on the efficiency of the equipment and process control. On the other hand, the equipment efficiency depends of their reliability and maintainability. In this work it is proposed a performance evaluation and analysis with the use of p-t- Timed Petri Nets using simulations, including process reliability analysis of the system through the throughput optimization, i.e., produced amount of goods. In this approach, a linear logic statement was developed and validated using a comparison of results of classes of states between the Tina simulation environment and the algorithm considered for a model of a producing – consuming system. A case study in the automotive industry is presented, consisting of the analysis of the real problems found in a body shop plant, with the use of Timed Petri Net.

Keywords: Timed Petri Nets. Flexible Manufacturing System. Modeling. Simulation. Performance Analysis.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SÍMBOLOS

$\wedge$	“e” lógico
$\vee$	“ou” lógico
$\omega_1 \Rightarrow \omega_2$	A palavra ω_1 gera (ou deriva) ω_2
+	Adição
Σ	Alfabeto
Σ^+	Alfabeto sem o símbolo vazio
K(*p*)	Capacidade do lugar *p*
$\Re$	Comjunto dos números reais
$\Re_+$	Comjunto dos números reais positivos
$\|\alpha\|$	Comprimento da palavra α
Σ^*	Concatenação dde alfabeto
Q	Conjunto de estados (finito)
Q_0	Conjunto de estados iniciais
[*M*>	Conjunto de marcações alcançáveis a partir de *M*
$\aleph$	Conjunto dos números naturais
$\aleph^+$	Conjunto dos números naturais sem o zero
Q^+	Conjunto dos números racionais positivos
$\varnothing$	Conjunto Vazio
$\subseteq$	Contido
$\neq$	Diferente
$\exists$	Existe
$\exists\|$	Existe ao menos um
=	Igual

∞	Infinito
$\cap$	Interseção entre conjuntos
$\geq$	Maior ou igual que
$>$	Maior que
$\rightarrow$	Mapeamento
M_k	Marcação atual
$M(p)$	Marcação do lugar p
M_0	Marcação inicial
M_{k+1}	Marcação sucessora
A	Matriz de inicidência
$\leq$	Menor ou igual que
$<$	Menor que
*	Multiplicação escalar
$\notin$	Não pertence
Λ	Palavra vazia
$\in$	Pertence
$\forall$	Qualquer
s*	Repetição do símbolo s
σ	Seqüências de disparo
-	Subtração
|	Tal que
M	Uma marcação qualquer
$\cup$	União entre conjuntos
K	Vetor coluna das capacidades dos lugares

v_k Vetor de habilitação

v_{kt} Vetor de habilitação para a Rede de Petri Temporizada

SUMÁRIO

1 Introdução

Até 1960, o mercado mundial era caracterizado pelo aumento quantitativo da produção onde praticamente tudo o que se produzia podia ser vendido. Nesta época, embora o preço fosse um fator importante para fomentar o aumento das vendas, a pressão que se exercia sobre este fator não era grande. Desde então, a evolução natural do sistema produtivo e a globalização puseram fim a esta fase dando origem a um período de mercados escassos, maior exigência de qualidade e maior competitividade. A competição por preços tornou-se acirrada em mercados emergentes, fazendo com que muitas empresas fossem reestruturadas, até mesmo mudassem de região ou país. O preço se tornou algo importante para o sucesso visto que os clientes podiam selecionar preços comparando produtos feitos em diversos países. No final dos anos 60, a relação produção/consumo mudou novamente, e os consumidores passaram a se questionar e a prestar mais atenção na qualidade dos produtos comprados, fazendo com que isto se tornasse um importante fator para o sucesso de uma empresa. Uma nova mudança se iniciou no final dos anos 70 onde os consumidores aumentaram bastante o poder de escolha e a capacidade das empresas aparentemente ainda excedia a demanda. As empresas tiveram que se modernizar, reduzir o intervalo de tempo entre o lançamento de novos produtos para poderem conquistar um público mais exigente em várias partes do mundo. A partir dos anos 90 passa a assumir a liderança um novo fator, isto é, a inovação, a habilidade de renovar rapidamente, no sentido do atendimento mais fiel aos requisitos (existentes ou emergentes) e não simples mudança superficial ou de estilo. É justamente neste período em que se encontra o atual mercado, onde é fundamental aderir aos ciclos da inovação para ter sucesso, caso contrário o fracasso é inevitável (JUNQUEIRA, 2001).

Assim, devido à alta competitividade no mundo globalizado de negócios, bem como a evolução do "*e-business*", são solicitados sistemas de produção de baixo custo, melhor qualidade, alta flexibilidade e confiabilidade ao longo da cadeia produtiva, de forma a manter a lucratividade necessária para a auto-sustentação dos negócios. Neste contexto a manufatura tem implantado a filosofia de produção *"Just-In-Time"* (JIT) que tem como base a eliminação do desperdício, redução do custo de produção, e a promoção do controle total da qualidade, sendo que seu objetivo

primário é produzir a quantidade certa, no lugar certo, e no tempo certo, com o mínimo de estoque e, de acordo com a demanda (NAKASHIMA, 2003). Neste âmbito, as montadoras de veículos devem estar preparadas para montar veículos personalizados tão rapidamente quanto possível e com o menor custo (MALINOWSKI, 2001).

A necessidade de aumento de produção, da redução de custos e do aumento da qualidade de bens de consumo, tem motivado a constante evolução dos sistemas de produção (ARAÚJO ET AL., 2001). Observa-se uma migração dos tradicionais sistemas de produção baseados em produção em lote para uma arquitetura em que se tem maior flexibilidade, caracterizando os sistemas modernos de manufatura.

Os sistemas modernos de manufatura são também chamados de Sistemas Flexíveis de Manufatura (SFMs). Os SFMs produzem uma variedade de produtos modificando a sua configuração de acordo com o planejamento da produção. Esta flexibilidade permite uma alocação mais rápida dos recursos, mas incrementa a complexidade de controle do sistema (NAKAMOTO, 2001). Dentro dos SFMs, vários eventos são esperados em um período de tempo futuro e, recursos devem ser alocadas em um certo período de tempo em relação às tarefas a serem executadas e as suas metas. Desta forma, o tempo é uma variável que deve ser considerada de duas formas: (1) qualitativamente, para a sincronização entre ações e eventos; (2) quantitativamente, para modelar a duração das ações com respeito às metas ou funções de custo (GHALLAB ET AL., 2004). Os SFMs são grupos de máquinas (robôs, transportadores, dispositivos, etc.) que trabalham em uma seqüência cíclica de atividades. A taxa de saída dos produtos de um SFM depende da seqüência de atividades nas máquinas, bem como da seqüência em que as diferentes peças são processadas por estas máquinas (ZUBEREK, 1996).

A performance corresponde a capacidade de um sistema de dar o resultado desejado, isto é, sua eficiência ou desempenho em atingir o planejado. Assim uma forma de mensurar a performance é avaliar a saída real do sistema em análise com sua respectiva saída planejada ou teórica. A performance dos SFMs depende da eficiência dos equipamentos (máquinas) e do controle do processo. A eficiência dos equipamentos depende de sua confiabilidade e manutenabilidade. A confiabilidade dos equipamentos está relacionada à idéia de confiança, durabilidade, segurança, credibilidade e disponibilidade para operar sem falhas. A confibilidade requerida do equipamento depende da confiabilidade operacional (operação e manutenção dos

mesmos) e a confiabilidade intríseca dos equipamentos. Uma das formas para se aumentar a eficiência dos equipamentos é reduzir o número de falhas inesperadas (i.e., paradas não programadas). Neste contexto, a confiabilidade dos equipamentos automatizados influencia na performance dos SFMs diretamente. Entretanto, equipamentos com alta confiabilidade são mais caros. Por outro lado, diagnósticos e correção de falhas mais rápidas (menor MTTR) podem aumentar a eficiência dos equipamentos (KUO & HUANG, 2000).

O objetivo principal dos Sistemas Flexíveis de Manufatura é seguir o planejamento geral do sistema que maximiza a saída e minimiza os estoques intermediários de modo a satisfazer as restrições econômicas (LEE & KORBAA, 2004).

O planejamento foca o problema do que deve ser feito. A programação foca o problema de como realizar um dado conjunto de ações usando um número limitado de recursos dentro de um espaço de tempo limitado. Uma programação é um conjunto de alocações de recursos e tempos iniciais para que as tarefas tenham disponíveis todos os recursos necessários e satisfaçam as restrições existentes. Basicamente, qualquer abordagem que seja feita para maximizar a saída, deve tratar 02 (dois) itens: gerar uma programação alternativa em que os conjuntos de tarefas sejam realizados no menor tempo possível e, avaliar esta programação (ZUBEREK, 1996).

Uma boa programação otimiza a função custo. Assim, o objetivo da programação é minimizar a função custo e o tempo necessário para a execução das tarefas (GHALLAB ET AL., 2004).

Dentro deste contexto, este trabalho enfoca o problema da avaliação da performance dos SFMs, através das Redes de Petri Temporizadas, incluindo a avaliação da confiabilidade do processo através da análise da quantidade total de itens produzidos na saída do sistema.

Nas últimas décadas, a modelagem, o gerenciamento e a análise dos SFMs têm recebido uma atenção especial dos pesquisadores e engenheiros. O crescimento da complexidade dos modernos SFMs do ponto de vista da produção, controle do processo, comunicação, etc., criou diversos problemas para o projeto e operação destes sistemas os quais requerem modelagem e análise para se determinar as condições de projeto e política de operação ótimas. Vários são os esforços da comunidade científica para estudar os Sistemas Flexíveis de

Manufatura, no que se refere a sua modelagem, o gerenciamento (ex.: regras de decisão) e a análise (ex.: otimização da performance de produção), entre outros temas.

Por exemplo, do ponto de vista da modelagem: Zhou et al. (1993), apresenta como alternativa a modelagem e análise dos SFMs utilizando Rede de Petri, bem como a análise do tempo de ciclo através da conversão da rede para um Grafo Marcado com o uso das técnicas de redução; Kuo & Huang (2000), utilizam a Rede de Petri Colorida para a modelagem, análise e diagnósticos de falha para os SFMs. Santos Filho et al. (2001) utilizam o E-MFG (*Enhanced Mark Flow Graph*) e o PFS (*Production Flow Schema*), isto é, interpretações da Rede de Petri no sentido de estabelecer uma metodologia para o projeto estruturado de sistemas de controle da produção; Boufaied (2002), estuda os processos de detecção de falha utilizando a combinação das Redes de Petri p-Temporizada e t-Temporizada para descrever através de crônicas as situações de falhas ou evoluções errôneas do processo. Di Febbraro (2002), realiza a modelagem de SFMs utilizando as equações de estado da Rede de Petri Temporizada, distinguindo duas classes de estados, as tangíveis e as intangíveis, e no qual a análise de performance é feita através da otimização da função custo, associada à permanência dos estados tangíveis. Saitou et al. (2002), utilizam Rede de Petri colorida para a modelagem dos recursos e o planejamento da produção em um sistema flexível de manufatura, onde se considera o objetivo de redução de custo, através do uso de um algoritmo genético que tem como regra o menor tempo de operação; Cassez & Roux (2003), mostram uma translação estruturada da Rede de Petri Temporizada para o Autômato Temporizado; Giua & Basile (2004), utilizam observadores de estado em uma Rede de Petri Temporizada para a recuperação de *deadlocks*. Hennemann et al. (2004), propuseram um modelo híbrido de sistema de apoio à decisão, utilizando-se simulação e Rede de Petri Colorida como técnica de modelagem; Mevius (2004) introduz um novo tipo de Rede de Petri de Alto Nível (Rede ML) para o controle da produção, onde a decisão tomada é baseada na comparação de certos indicadores com suas metas, periodicamente ou continuamente analisadas; Zhu et al. (2004), apresentam um novo método de programação pela combinação da Rede de Petri Temporizada com a Álgebra *Max Plus*, onde as funções podem ser usadas para o planejamento e controle dos SFMs em tempo real; Marin et al. (2005), propõem o uso de redes Fuzzy-Neurais juntamente com a Rede de Petri em tempo Real, formando a

FNRTPN (*Fuzzy Neural Real Time Petri Net*), entretanto, esta abordagem depende do conhecimento de todos os estados do sistema modelado para se poder avaliar a disponibilidade dos recursos e controlar as tarefas através do controle dos disparos das transições, por outro lado, a arquitetura não possibilita a análise do processo. Tsinarakis et al. (2006), realizam a modelagem, análise e síntese utilizando a Redes de Petri Temporizada Híbrida e, efetuam a análise de performance através de simulação. Do ponto de vista do gerenciamento: Juia & Vallete (2000), abordam uma simulação em tempo real utilizando Rede de Petri Temporizada (*p-time*) para a programação em tempo real de sistemas de produção em lote; Simão et al. (2001), através da abordagem dos sistemas de manufatura flexíveis orientados a eventos, estabelecem um processo de decisão realizado por um conjunto de agentes, que se orientam através dos estados discretos obtidos na monitoração, implementando o comportamento de regras condição/ação, isto é, a decisão é realizada por um conjunto de regras que expressam, por meio de relações causais, como se dá a transição de estados no sistema, ou ainda, por um conjunto de regras que se apóiam nos fatos. Desta forma, as “regras” interagem com a “base de fatos” avaliando estados e alterando seus valores; Malinowski (2001), utiliza um algoritmo genético para a otimização da seqüência na qual veículos são colocados na linha de montagem, considerando o tipo do veículo e os seus opcionais; Fanti (2003), propõe uma estratégia de controle para gerenciar os recursos em sistemas que permitem o compartilhamento dos mesmos com a finalidade de prevenir os *deadlocks* utilizando a modelagem por Rede de Petri Colorida; Maia et al. (2004), utilizam um sistema de planejamento da produção baseado em simulação (PPSS - *Production Planning System based on Simulation*) de modo a avaliar decisões de programação de suprimento, como uma maneira de reduzir custos globais de estocagem, além de alcançar a produtividade máxima ao evitar falta de suprimento. Do ponto de vista da análise: Zuberek (1996), estuda a otimização de programação através da análise de invariantes da Rede de Petri Temporizada; Jeng et al. (2000), analisam a performance de sistemas de manufatura de semicondutores utilizando a Rede de Petri Temporizada Markoviana, uma subclasse da rede estocástica generalizada (GSPN); Nakashima (2003), utiliza a Rede de Petri Estocástica Generalizada para a análise de sistemas de logística; Chauvet et al. (2003), utilizam um método heurístico para a otimização de uma produção cíclica, pela minimização do estoque intermediário através da otimização do gargalo produtivo, mostrando que esta

condição é quase ótima, senão ótima; Giua & Basile (2004), mostram o uso de um observador de estado para estimar o estado da planta, isto é, as marcas de uma Rede de Petri. A estimação é feita através da observação das palavras de eventos, isto é, disparo das transições; Bernardi & Campos (2004), estudam a performance dos sistemas utilizando Redes de Petri t-Temporizada através da análise das propriedades estruturais; Lee & Korbaa (2004) analisaram o planejamento cíclico da produção para a determinação do tempo ótimo e a minimização do estoque intermediário, bem como o mix utilizando Rede de Petri Temporizada; Lee & DiCesare (2004), utilizam um método de heurística, modelado em Rede de Petri, para melhorar a desempenho do uso de VATs (Veículos Autônomos de Transporte) na logística, através da árvore de alcançabilidade da rede e otimização dos custos de forma global; Bucci et al. (2005), analisam a performance de sistemas em tempo real utilizando Rede de Petri Estocástica através de uma técnica de análise dos tempos e das seqüências dos eventos.

Entretanto, a análise de performance dos sistemas é feita sem levar em consideração a ocorrência de falhas, e sem analisar a influência das falhas na quantidade de itens produzidos na saída do sistema, resultando em tomadas de decisão que nem sempre favorecem o melhor desempenho e menores custos dos sistemas. Na prática, quando os sistemas entram em produção e suas características diferem das características utilizadas durante a fase do projeto, o planejamento da produção passa a ter mais um problema: saber qual a real capacidade do sistema e, desta forma, poder avaliar a performance do sistema em relação a sua real capacidade e, também, em relação a capacidade do projeto. A proposta deste trabalho é justamente prover um método de modelagem utilizando a Rede de Petri Temporizada e considerando as ocorrências de falhas (levando-se em consideração as taxas de falhas constantes) que possibilite resolver o probelma da incerteza sobre a real capacidade de produção do sistema e, a partir daí, analisar a performance dos mesmos sob diversos pontos de vista ou situações que ocorrem no dia-à-dia no gerenciamento dos SFMs.

1.1 Objetivos

O objetivo do presente trabalho é estudar e desenvolver um método para a análise de performance dos SFMs, utilizando a Rede de Petri Temporizada e, levando-se em consideração a confiabilidade dos equipamentos e dos processos na modelagem dos sistemas, numa linguagem mais próxima à especificação dos requisitos mais usuais de confiabilidade, isto é, a utilização de taxas de falhas constante, que são comumente usadas por deficiência no conhecimento das características dos equipamentos ou processos, ou ainda, pela falta usual de histórico de falhas dos mesmos. A análise de performance é feita através da quantidade dos *buffers*, disponibilidade dos equipamentos, e quantidade de itens na saída do sistema.

1.2 Organização do Texto

No capítulo 2, são apresentados uma revisão bibliográfica sobre os conceitos fundamentais utilizados no trabalho, tais como, sistemas flexíveis de manufatura, teoria dos autômatos finitos, autômatos temporizados, Rede de Petri, Rede de Petri Temporizadas e, as justificativas para a escolha da ferramenta de simulação. No capítulo 3, são apresentados a descrição da ferramenta de simulação escolhida, uma proposta para a abordagem da confiabilidade dos recursos e processos com a Rede de Petri Temporizada, o desenvolvimento de um exemplo simples de aplicação da proposta apresentada e, sua aplicação em um caso real encontrado na indústria automobilística. No capítulo 4, é apresentado em detalhes um estudo de caso com o exemplo real apresentado no capítulo anterior incluindo-se os resultados das simulações realizadas. Finalmente, no capítulo 5, são apresentados os resultados que são comentados e analisados e, bem como as propostas para trabalhos futuros.

2 Revisão Bibliográfica

2.1 Sistemas Flexíveis de Manufatura

Os Sistemas Modernos de Manufatura são caracterizados por um baixo custo de produção, alta qualidade, alta flexibilidade e, pela produção na quantidade certa, como mínimo estoque e de acordo com a demanda (NAKASHIMA, 2003). Os sistemas que satisfazem todas estas características são chamados de Sistemas Flexíveis de Manufatura (SFMs). Existem diversos tipos de flexibilidade em um SFM (CHANDRA et al., 2005): flexibilidade de expansão, de volume, de modificação, de inserção de novos produtos e de alteração no *mix* de produção.

Os SFMs produzem uma variedade de produtos modificando as suas configurações de acordo com o planejamento da produção. Esta flexibilidade permite uma alocação rápida dos recursos, mas incrementa a complexidade de controle do sistema (NAKAMOTO, 2001). Desta forma, equipamentos autônomos, cooperando entre si através de comunicação via redes industriais, aliados a um fluxo contínuo de diferentes produtos pelas linhas de produção, tornam as plantas de produção cada vez mais difíceis de serem projetadas, especificadas e administradas (ARAÚJO ET AL., 2001). Isto é, a modelagem, o planejamento (seleção do tipo de peça), a programação e o controle (gerenciamento dos recursos, principalmente os compartilhados) são os principais problemas encontrados nos SFMs (SAYGIN & KILIC, 2004).

Segundo (SHIM & SIEGEL, 1999), os SFMs são definidos como sistemas de produção controlados por computador com tecnologia suficientemente adequada para produzir uma variedade moderada de produtos dentro de um volume moderado e flexível.

Os SFMs são compostos de duas partes principais: o sistema físico (ou sistema de controle) e o sistema de gerenciamento (ou sistema de decisão) (DICESARE & HARHALAKIS, 1993). Esta hierarquia é representada através da Figura 2.1 (JAIN & FOLLEY, 2002), onde a planta produtiva representa o sistema físico e, o sistema de gerenciamento é composto pelos blocos do nível de controle

de operação e o nível de planejamento dos SFMs, camda na qual se encontra o sistema de decisão.

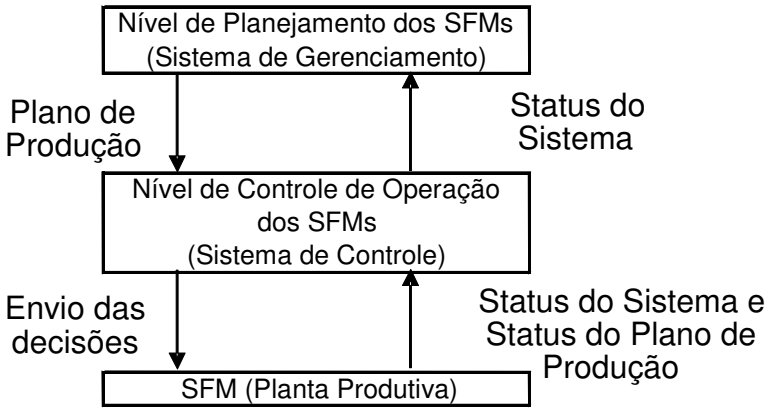

Figura 2.1: Hierarquia do Sistema de Controle de um SFM.

O objetivo principal dos Sistemas Flexíveis de Manufatura é atender o planejamento geral do sistema maximizando a saída e minimizando os estoques intermediários de modo a satisfazer as restrições econômicas (LEE & KORBAA, 2004). Entretanto, em geral, o planejamento dos sistemas é feito tendo como premissas básicas o sequenciamento estático das tarefas, tempos de operação determinísticos, e nenhuma falha de máquina ou de logística (carregamento de peças) o que é incompatível com a realidade, onde as interrupções não previstas ocorrem, e conseqüentemente, têm-se perdas no sistema produtivo.

Na ocorrência de uma interrupção não prevista, o nível operacional deve reagir e gerenciar os recursos de modo a minimizar os efeitos sobre o desempenho da produção. Neste processo de decisão algumas questões precisam ser resolvidas (Jain & Folley, 2002):

- Como é quantificada a interrupção?
- Qual é o efeito da interrupção sobre o desempenho da produção no cumprimento do plano?
- Qual é o nível de impacto sobre o plano e produção no qual deveria ter sido tomada uma decisão de reação?
- Como será possível determinar se a modificação do plano de produção é possível de ser seguido?
- Como é o plano de produção modificado?

- Qual é o efeito da modificação do plano de produção sobre a performance do sistema?

O gerenciamento dos recursos é feito através da tomada de decisão, imediata ou não, após a análise da situação do sistema produtivo, o que implica que quanto mais rápida for a análise, mais rapidamente as decisões podem ser tomadas, e mais rapidamente os desvios podem ser controlados.

Se a alocação de recursos compartilhados não for gerenciada adequadamente, é possível ocorrer um autotravamento ou "*deadlock*" do sistema. O "*deadlock*" do sistema ocorre quando o fluxo dos processos é permanentemente impedido quando, então, as operações nos processos não podem mais ser executadas (NAKAMOTO, 2001).

O travamento de um sistema produtivo pode ser detectado pelo *status* dos equipamentos, quando da ocorrência dos estados "*blocked*" (quando o equipamento não pode continuar o ciclo produtivo devido a impossibilidade de retirada da peça já processada devido a impossibilidade de se alocar recursos no processo seguinte) ou "*starved*" (quando o equipamento não pode continuar o ciclo produtivo devido à falta de recursos para a alimentação do processo). Tais estados podem ser conseqüência de falhas nos equipamentos, falta de mão-de-obra, falha no processo (qualidade), falha dos operadores, etc.. Como descrito em Cho (1993), pode-se ter três tipos de "*deadlock*" no sistema produtivo:

1. *Deadlock* de fluxo de processos ("*Part Flow Deadlock*")
2. *Deadlock* de recursos ("*Processing Resource Deadlock*")
3. *Deadlock* de materiais ("*Material Handling Deadlock*")

2.1.1 Modelagem e Simulação dos Sistemas Flexíveis de Manufatura

Os processos em sistemas de manufatura são paralelos e distribuídos, e necessitam de análises qualitativas (ausência de *deadlock*, ausência de *overflows*, mútua exclusão em recursos compartilhados), e quantitativas (propriedades de performance, propriedades de tempo de resposta, etc.) (DICESARE & HARHALAKIS, 1993).

A Rede de Petri apresenta um formalismo matemático, e pode ser considerada como uma ferramenta gráfica especialmente conveniente para

modelar e analisar (de forma qualitativa e quantitativa) sistemas dinâmicos a eventos discretos os quais exibem processos concorrentes e são caracterizados pelos fenômenos de sincronização de tarefas e compartilhamento de recursos, assim como os Sistemas de Manufatura (DICESARE & HARHALAKIS, 1993). Uma das maiores vantagens na modelagem com Rede de Petri é que o mesmo modelo pode ser usado tanto para a análise das propriedades comportamentais e análise de performance, bem como para uma construção sistemática de simuladores e controladores a eventos discretos (ZURAWSKI & ZHOU, 1994). Desde sua introdução, as Redes de Petri têm sido amplamente usadas para a modelagem e estudos de sistemas concorrentes, assíncronos, distribuídos, paralelos, estocásticos e com compartilhamento de recursos, como por exemplo, os sistemas de manufatura.

Existem também outras ferramentas que podem ser utilizadas na análise dos sistemas discretos (incluindo a análise de performance) como os programas de simulação (ProModel, Arena, AutoMod, etc.), os quais estão embasados na teoria das Redes de Filas (KOBAYASHI, 1978), (LAVENBERG, 1983). Porém estas ferramentas possuem restrições para a conversão dos modelos em especificações de estratégias de controle porque envolvem a abstração de conceitos e decisões que não podem ser diretamente implementados (JUNQUEIRA, 2001).

O uso da Rede de Petri para modelagem, análise e controle de sistemas a eventos discretos, deve–se às seguintes razões (ZHOU ET AL., 1992):

- Rede de Petri possui uma representação gráfica de fácil entendimento e diretamente relata características chaves de controle de eventos discretos em um sistema de manufatura.
- Rede de Petri possui uma fundamentação matemática para análise de consistência lógica.
- Rede de Petri exibe propriedades de decomposição (refinamento) tornando possíveis os projetos modulares.
- É possível traduzir ou compilar a rede em um código de controle ou dados para execução e implementação no chão-de-fábrica.

Por vezes a grande vantagem da representação gráfica não é perceptível quando utilizada para a representação de sistemas complexos, como os sistemas flexíveis de manufatura.

Neste contexto, baseado em diferentes definições e interpretações para a modelagem estática e dinâmica dos componentes da rede, tipos diferentes de Rede de Petri têm sido derivadas, como por exemlo, as chamadas de Redes de Petri de alto nível (Ex.: Redes Predicado/Transição, Redes de Petri Coloridas, Redes de Petri Orientada a Objetos), as quais têm mostrado que são convenientes para a modelagem de sistemas complexos (MEVIUS, 2004).

O desenvolvimento de SFMs requer uma análise funcional e de performance para a verificação e validação dos requisitos operacionais do sistema. Rede de Petri, como uma ferramenta matemática, permite a análise de performance dos sistemas modelados. Tanto para os sistemas determinísticos quanto para os sistemas estocásticos a medição da performance pode ser realizada através das funções de tempo definidas, utilizando técnicas analíticas ou por meio de simulação (ZURAWSKI & ZHOU, 1994).

Para estudar os aspectos de performance dos modelos em Rede de Petri, a duração das atividades deve ser levada em consideração dentro das especificações do modelo. Vários tipos de Rede de Petri “com tempo” têm sido propostos para associar tempos de disparos aos lugares ou transições das redes. Na rede estocástica, os disparos das transições são eventos instantâneos, assim como nas redes ordinárias (i.e, “sem tempo”), entretanto, as fichas são atrasadas nos lugares por períodos de tempo (distribuídos exponencialmente), determinados pelas transições conectadas ao lugar. Na rede temporizada, os disparos das transições são eventos em tempo real, ou seja, fichas são removidas dos lugares de entrada no início do período de disparo, e são depositadas nos lugares de saída ao final deste período (ZUBEREK & KUBIAK, 1994). Geralmente, a análise de performance dos SFMs tem sido realizada na literatura através de técnicas analíticas, por meio da análise dos invariantes das classes de estado, ou através das técnicas de redução, ou ainda através de algoritmos heurísticos. Entretanto, devido a complexidade dos modelos de SFMs o uso de tais técnicas pode ser proibitivo e, a técnica de simulação a eventos discretos passa a ser então a única técnica viável para a análise de performance (ZURAWSKI & ZHOU, 1994).

Para aumentar a performance nos SFMs, alguns desafios, existentes nos dias atuais, precisam ser minimizados ou eliminados. Os maiores desafios são:

- Informações de suporte para a tomada de decisões, usadas pelos líderes da produção, deficientes;
- Informação sobre a produção limitada devido a falta de monitoramento e previsibilidade da expectativa de produção e impactos financeiros;
- Inconsistência na análise e interpretação dos dados obtidos em tempo real sobre o comportamento futuro do sistema;
- Quando existentes, a precisão e o tempo de resposta das análises sobre a expectativa da performance do sistema limitam a tomada de decisões.

Por outro lado, o princípio geral de planejamento em tempo real de um sistema de produção é baseado na simulação em tempo real, a qual é feita em paralelo com a produção em tempo real do sistema em questão (Julia & Valette, 2000), onde se verifica mais uma vez a importância da modelagem e simulação do sistema utilizando-se a Rede de Petri Temporizada, haja visto a necessidade de previsão do comportamento e performance do sistema no processo de tomada de decisão do ponto de vista quantitativo e de forma objetiva baseando-se nos fatos ocorridos e na provável situação futura esperada a ser gerenciada.

Somente por meio da modelagem e simulação é possível responder assertivamente, antecipadamente e quantitativamente, sobre a performance a ser obtida pelo sistema, levando-se em consideração que a quantificação da interrupção não prevista e seus efeitos podem ser mensurados, o que possibilita a tomada de decisão e a execução de ações sobre o sistema (planejamento ou execução).

O comportamento típico do funcionamento de supervisão de um sistema produtivo realimentado ou em malha fechada é apresentado no diagrama de blocos da Figura 2.2. Neste âmbito, os sistemas utilizam técnicas que permitem, através do uso dos autômatos, acompanhar (ou “observar”) a performance do sistema a partir dos eventos ocorridos e, com o uso de controladores e algoritmos de controle, se necessário, tomar alguma ação corretiva (“reação”), após a análise e diagnóstico da situação. Entretanto, os supervisórios somente atuam do ponto de vista passado, isto é, o monitoramento do instante presente para trás. Por esta razão, a modelagem do sistema se faz necessária para a atuação antecipada ou do ponto de vista futuro, onde de forma análoga, através

do resultado das simulações obtidos dos modelos desenhados, é possível tomar alguma ação corretiva (ou preventiva), se necessário.

Sob esta ótica de controle em mala fechada, a Rede de Petri é extremamente poderosa para a modelagem dos sistemas devido a possibilidade de simulação em tempo real e a sua estreita relação com a teoria dos autômatos, descrita no Capítulo 2.

Os sistemas supervisórios têm se mostrado de fundamental importância na estrutura de gestão das empresas, fato pelo qual deixaram de ser vistos como mera ferramentas operacionais, ou de engenharia, e passaram a ser vistos como uma relevante fonte de informação. Os sistemas de supervisão de processos industriais automatizados desempenham três atividades básicas (UDDIN ET AL., 2000): supervisão, operação e controle.

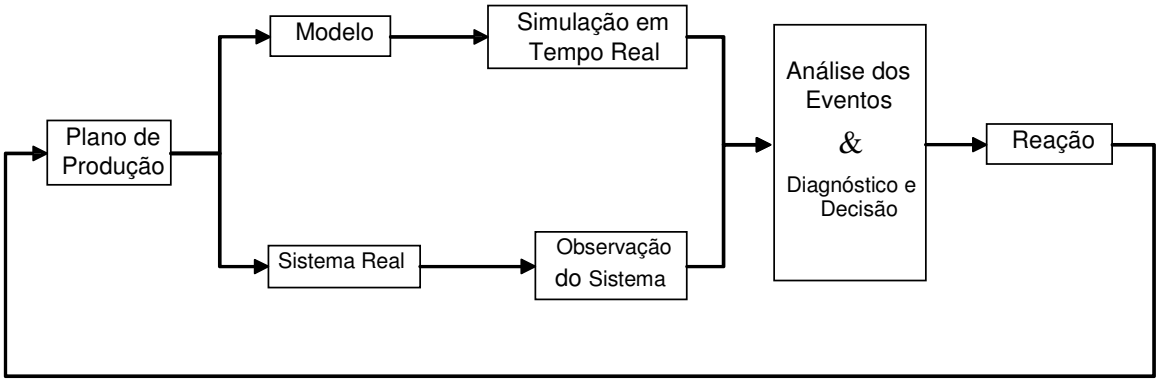

Figura 2.2: Diagrama de Blocos de uma Estrutura de Planejamento em Tempo Real de um Sistema de Produção.

Todavia, na prática as empresas têm deixado em segundo plano a modelagem dos sistemas, são raros os casos em que a modelagem é encontrada e, quando feitas não possuem a abordagem necessária para a representação e caracterização dos sistemas (como por exemplo a variável tempo e as características de confiabilidade dos equipamentos), tendo como resultado a incapacidade de gestão do sistema produtivo, bem como sua performance.

2.2 Conceitos Fundamentais

Uma Rede de Petri pode ser visto como geradora de uma linguagem (GIUIA, 1991), onde o conjunto de alcançabilidade da rede é equivalente à linguagem gerada pela rede. Além disso, a representação dos sistemas a partir das linguagens permite uma otimização dos modelos utilizados e a especificação da síntese automática da rede a partir da especificação do comportamento (definição da tarefa) desejado do sistema como linguagem (RESTREPO, 2004).

Por outro lado, os autômatos finitos temporizados têm sido utilizados para a modelagem do comportamento dos sistemas de tempo real, promovendo a partir da perspectiva da teoria formal das linguagens a análise das propriedades do sistema (ALUR & DILL, 1994).

Assim, neste capítulo apresenta-se a teoria formal de Linguagens, Autômatos Temporizados e, a Rede de Petri Temporizada.

2.2.1 Linguagens

Na teoria das Linguagens Formais (COHEN, 1996), define-se como **alfabeto**, o conjunto das unidades fundamentais (símbolos) com as quais são construídas as estruturas. Símbolo ou Caractere é uma unidade básica. Letras e dígitos são exemplos de símbolos. Geralmente, um alfabeto é representado pela letra grega maiúscula $\sum$. **Palavras** são definidas como um conjunto de símbolos justapostos provenientes de um alfabeto, e que representam eventos. **Linguagens** são definidas como a concatenação de Palavras com ou sem repetição e que representam as tarefas. A concatenação finita de símbolos é definida por

$$cat_{\Sigma} = \Sigma \times \Sigma \rightarrow \Sigma^{*} \tag{1}$$

Em que $\sum^*$ é o conjunto (finito) de todas as palavras construídas com os símbolos de $\sum$.

A concatenação é uma operação definida sobre uma linguagem, a qual associa a cada par de palavras, uma palavra formada pela justaposição da primeira com a segunda. A concatenação possui as propriedades de associatividade e de elemento neutro à esquerda e à direita.

Seja *S* um conjunto de palavras, então, definimos *S** como sendo o conjunto de todas as palavras formadas pela concatenação das palavras de *S*, onde qualquer palavra pode ser repetida quantas vezes se queira, e onde a palavra vazia é também inclusa.

A palavra vazia ou nula é denotada por Λ.

O comprimento de uma palavra, representado por |*s*|, é igual ao número de símbolos que a compõe. A palavra vazia é a única palavra de comprimento nulo, isto é, $|\Lambda| = 0$.

Teorema: Para qualquer conjunto *S* de palavras, temos que *S** = *S***.

A concatenação de linguagens é a operação $cat_S = \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$, onde:

$$cat_S(\{\Lambda\}; S) = cat_S(S; \{\Lambda\}) = S;\ S \subset \Sigma^* \tag{2}$$

$$cat_S = (S_1; S_2) = \{s_1 s_2 \mid s_1 \in S_1 \wedge s_2 \in S_2\} \tag{3}$$

Exemplo: Dadas as linguagens $S_1 = \{a; ab\}$ e $S_2 = \{c; de\}$, tem-se:

$$cat_S(\{\Lambda\}; S_1) = cat_S(S_1; \{\Lambda\}) = S_1 = \{a; ab\} \tag{4}$$

$$cat_S(\{\Lambda\}; S_2) = cat_S(S_2; \{\Lambda\}) = S_2 = \{c; de\} \tag{5}$$

$$cat_S(S_1; S_2) = \{ac; ade; abc; abde\} \tag{6}$$

$$cat_S(S_2; S_1) = \{ca; cab; dea; deab\} \tag{7}$$

Dada duas linguagens $S_1; S_2 \in \Sigma^*$, a operação de união é definida por:

$$S_1 \cup S_2 = \{s \in S_1 \vee s \in S_2\} \tag{8}$$

Dada duas linguagens $S_1; S_2 \in \Sigma^*$, a operação de intersecção é definida por:

$$S_1 \cup S_2 = \{s \in S_1 \wedge s \in S_2\} \tag{9}$$

Exemplo: Se $\Sigma = \{a; b; c\}$ é o alfabeto e $S_1 = \{\Lambda; a; caa\}$ e $S_2 = \{b\}$, então a união de S_1 e S_2 é

$$S_1 \cup S_2 = \{\Lambda; a; caa; b\} \tag{10}$$

E a intersecção de S_1 e S_2 é

$$S_1 \cap S_2 = \{\ \} \tag{11}$$

2.2.2 Gramática

Informalmente, uma gramática é uma enumeração ou conjunto de leis de formação de linguagens. Uma gramática serve para definir qual o subconjunto de

palavras que faz parte de uma determinada linguagem. Ela é um dispositivo formal para especificar uma linguagem potencialmente infinita de uma forma finita.

Formalmente, uma gramática *G* é uma quádrupla $G = (N, T, P, S)$, onde:

- *N* é o conjunto finito de não-terminais;
- *T* é o conjunto finito de terminais;
  - Onde, $N \cap T = \phi$, $\Sigma = N \cup T$ em que Σ é um alfabeto.
- *P* é o conjunto finito de regras de produção, isto é, $P = \{\alpha \to \beta | \alpha \in \Sigma^+ \wedge \beta \in \Sigma^*\}$, e $\Sigma^+ = \Sigma^* - |\Lambda|$.
- *S* é o conjunto de palavras inicial da gramática.

2.2.2.1 Linguagem gerada por uma Gramática

Dada uma gramática *G*, a linguagem *L* gerada por *G*, algumas vezes denotada *L(G)*, é o conjunto $L = \left\{ \omega \in T^* \middle| S \overset{*}{\Rightarrow} \omega \right\}$, ou seja, *L* é o conjunto de todas as cadeias de terminais geradas pelo símbolo inicial.

Linguagens derivadas de gramáticas são chamadas de **linguagens formais**.

Definição: Gerações (Derivações) em uma linguagem

- Geração (ou derivação) é uma operação de substituição efetuada de acordo com as regras de produção da gramática.
- Seja G uma gramática, $G = (N, T, P, S)$, e sejam ω_1 e ω_2 palavras formadas por Σ. Dizemos que ω_1 gera diretamente (ou deriva diretamente) ω_2, e denotamos $\omega_1 \Rightarrow \omega_2$, se $\alpha \to \beta$ é uma produção de G, ω_1 contém *α* como caso particular, e ω_2 é obtida de ω_1 substituindo-se como caso particular *α* por *β*. Se $\omega_1, \omega_2, ..., \omega_{n-1} \Rightarrow \omega_n$, dizemos que ω_1 gera (ou deriva) ω_n e denotamos por $\omega_1 \overset{*}{\Rightarrow} \omega_n$.
- A linguagem gerada por uma gramática, $G = (N, T, P, S)$, é o conjunto de todas as palavras da linguagem obtida da gramática por derivações sucessivas. Isto é, $L(G) = \{\omega | \omega \in T^* \wedge S \Rightarrow^* \omega\}$.

Segundo a hierarquia de Chomsky (HOPCROFT, 2000) existem quatro tipos de gramáticas:

- Tipo 0 ou Irrestristas: é a gramática que gera linguagens estruturadas em frase. Formalmente, $P = \{\alpha \to \beta | \alpha \in \Sigma^+, \beta \in \Sigma^*\}$;
- Tipo 1 ou Sensível ao Contexto: é a gramática que gera linguagens sensíveis ao contexto. Uma gramática é sensível ao contexto se, para toda a produção $\alpha \to \beta$, a palavra *β* é pelo menos tão longa quanto a palavra *α*, isto é, $|\alpha| \leq |\beta|$;
- Tipo 2 ou Livre de Contexto: é a gramática que gera linguagens livre de contexto. Uma gramática é livre de contexto se, para toda a produção $\alpha \to \beta$, *α* é uma única palavra não terminal, isto é, $P = \{\alpha \to \beta | \alpha \in N \wedge \beta \neq \Lambda\}$;
- Tipo 3 ou Regular: é a gramática que gera linguagens regulares. Uma gramática é regular se, para toda a produção $\alpha \to \beta$, *α* é uma única palavra não terminal e *β* é da forma *c* ou *cW*, onde *c* é um símbolo terminal e *W* é um símbolo não terminal.

A Figura 2.3 representa a Hierarquia de Chomsky na teoria de conjuntos para os tipo de gramática citados, onde a gramática Tipo 0 é a mais expressiva.

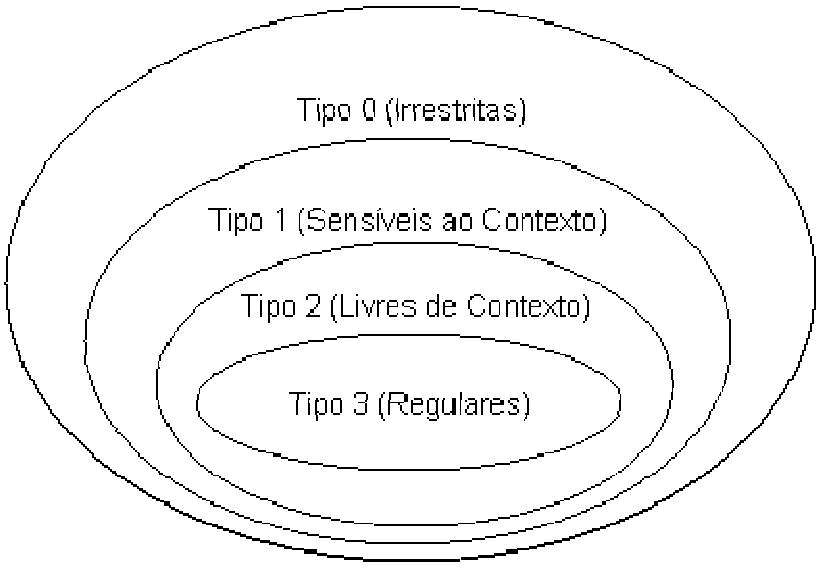

Figura 2.3: Tipos de Gramática segundo a Hierarquia de Chomsky.

2.2.3 Expressões Regulares

No estudo das linguagens formais, algumas linguagens podem ser representadas através de uma expressão regular, utilizando a álgebra convencional e os símbolos de um alfabeto. Assim, linguagens complexas podem ser representadas em termos de expressões simples (COHEN, 1996), (MONTGOMERY, 2004):

- a^* : representa a repetição do símbolo a, por um número arbitrário de vezes;
- w^* : representa a repetição da palavra *w*, por um número arbitrário de vezes;
- + : símbolo empregado como operador lógico *ou*, indicando uma opção entre duas ou mais possibilidades. Ou ainda, o símbolo utilizado para representar uma operação de concatenação, dado um símbolo a, $a^+ = a^*a = aa^*$.

A construção de expressões regulares segue as seguintes regras básicas:

i. $\varnothing$ é uma expressão regular denotando o conjunto vazio { }; *a* é uma expressão regular denotando o conjunto {*a*} para todo $a \in \Sigma$.
ii. Se *r* e *s* são expressões regulares, então *rs*, $(r + s)^*$, r^* e s^* também são expressões regulares.
iii. Toda expressão regular é construída por meio da aplicação das regras i e ii, acima, um número finito de vezes.
iv. A palavra vazia Λ e a linguagem vazia $\varnothing$, também são consideradas nas expressões regulares para as quais se têm as seguintes propriedades:
    a. $\Lambda s = s\Lambda = s$
    b. $\Lambda^* = \Lambda$
    c. $\varnothing + L = L$
    d. $\varnothing L = L\varnothing = \varnothing$
    e. $\varnothing^* = \Lambda$

Qualquer linguagem que pode ser denotada por uma expressão regular é denominada de Linguagem Regular.

2.2.4 Autômatos Finitos

Um autômato é um dispositivo que pode reconhecer uma determinada linguagem, possuindo estados que representam as situações do sistema (MONTGOMERY, 2004). Se o número de estados é finito, o autômato é dito finito, caso contrário, o autômato é dito infinito.

Desta forma o autômato finito é uma coleção de três coisas:

1. Um conjunto finito de estados, um dos quais é designado como estado inicial ou de partida, e alguns (ou talvez nenhum) designados como estados finais.
2. Um alfabeto $\sum$ de possíveis símbolos de entrada, a partir dos quais são formadas palavras.
3. Um conjunto finito de transições as que denotam mudança de estado de acordo com cada símbolo do alfabeto de entrada.

O autômato reconhece uma determinada linguagem através da leitura seqüencial dos símbolos, mudando de estado a cada leitura. Assim, uma palavra é dita reconhecida se o estado alcançado após a leitura do último símbolo da palavra pertence ao conjunto de estados finais. Assim, cada símbolo lido atualiza o estado do autônomo de acordo com uma função de transição. Se esta função de transição leva o autômato a mais de um estado quando um certo símbolo é lido, o autômato é dito não determinístico, caso contrário, o autômato é dito determinístico.

Um autômato determinístico finito, ou simplesmente um autômato é uma quíntupla

$$A = (Q;\Sigma;\delta;q_0;Q_m) \tag{12}$$

em que:

- $Q = \{q_0;\cdots;q_n\}$ é um conjunto finito de estados;
- $\Sigma = \{\sigma_1;\cdots;\sigma_m\}$ é o alfabeto ou conjunto de símbolos;
- $\delta : Q \times \Sigma \times Q \to Q$ é a função de transição de estados, em que $\delta(q,\Lambda;q) = q$ e $\delta(q_i,\sigma;q_j) = \{(q_i,\sigma;q_j) \mid \forall q_i, q_j \in Q \; e \; \sigma \in \Sigma$;

- $q_0 \in Q$ é o estado inicial;
- $Q_m \subseteq Q$ é o conjunto de estados marcados como estados finais;

Observando a função de transição de estados δ, vê-se que *q* somente será um estado do autômato *A*, se o símbolo σ for uma entrada aceita por ele.

Graficamente um autômato pode ser representado por um diagrama de transição de estados, que é um grafo direcionado ou grafo de transição, onde os vértices são os estados e os arcos representam as funções de transição.

O grafo de transição foi inventado em 1957 por John Myhill. Um grafo de transição, abreviado por GT, é uma coleção de três coisas:

1. Um conjunto finito de estados, um dos quais é designado como estado inicial ou de partida (-), e alguns (ou talvez nenhum) designados como estados finais (+).
2. Um alfabeto $\sum$ de possíveis símbolos de entrada (*labels* ou subscrições dos arcos)
3. Um conjunto finito de transições (arcos) que mostram como ir de um estado a outro baseado na leitura dos símbolos de entrada especificados como *labels* (incluindo também o símbolo nulo Λ).

Formalmente um grafo de transição é uma n-upla dada por $G = (Q; E; \delta; \alpha; \beta)$, onde:

- $Q = \{q_0; \cdots; q_n\}$ é um conjunto finito de estados;
- *E* é um conjunto de eventos
- δ é um conjunto finito de transições;
- α e β são duas aplicações de δ em Q, no qual α é o estado inicial da transição e β é o estado final da transição;

Da mesma forma, um grafo de transição originado por um alfabeto Γ é uma n-upla dada por $G = (Q; E; \delta; \alpha; \beta; \Gamma)$, onde:

- $(Q; E; \delta; \alpha; \beta)$ é um grafo de transição;
- $\Gamma = \{\sigma_1; \cdots; \sigma_m\}$ é o alfabeto ou conjunto de símbolos;
- Γ é uma aplicação de δ em Q, que associa uma transição δ em um símbolo do alfabeto;

Um grafo de transição é finito se *Q* e *E* são finitos. Um grafo de transição é chamado determinístico se para todo estado *q* e cada símbolo *a*, existe ou pode existir um e somente um estado *q´* atingível a partir de *a*, caso contrário o grafo de transição é dito não determinístico. Um grafo de transição deve satisfazer aos seguintes axiomas (CORTADELLA, 1998):

1. Não existências de laços (*self-loops*);
2. Para todo o evento existe uma ocorrência e, portanto, existe um estado de origem;
3. Todo estado é alcançável a partir de um estado inicial;

Dessa forma, um autômato definido por $A = (Q;\Sigma;\delta;q_0;Q_m)$ pode ser representado graficamente pelo grafo de transição *G*, onde $q_0 = \alpha$ e $Q_m = \beta$.

Esta situação de representação é baseada no Teorema de Kleene.

Teorema de Kleene: Qualquer linguagem regular pode ser definida por um autômato finito, ou um grafo de transição, implicando em dizer que (COHEN, 1996):

1. Toda linguagem que pode ser definida por autômato finito também pode ser definida por um grafo de transição;
2. Toda linguagem que pode ser definida por um grafo de transição também pode ser definida por uma expressão regular;
3. Toda linguagem definida por uma expressão regular também pode ser definida por um autômato finito;

Exemplo:

O autômato representado graficamente na Figura 2.4 é definido por:

- $\Sigma = (\alpha;\beta;\gamma)$
- *Q=(0;1;2)*
- $\delta(\alpha;0) = 1,\ \delta(\alpha;1) = 2,\ \delta(\alpha;2) = 2,\ \delta(\beta;0) = 2,\ \delta(\beta;1) = 0,\ \delta(\beta;2) = 1,$ $\delta(\gamma;1) = 1,\ \delta(\gamma;2) = 0;$
- $q_0 = 0$ e
- $Q_m = \{1;2\}$

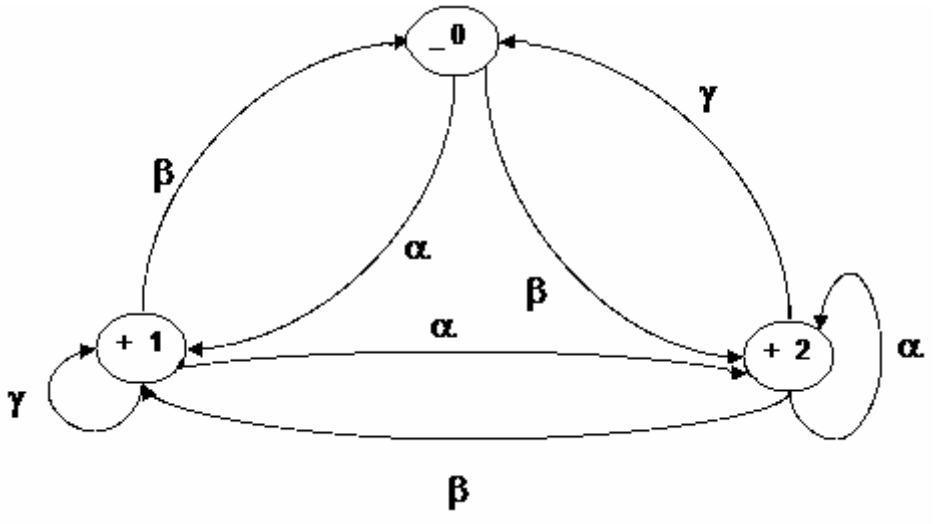

Figura 2.4: Exemplo de autômato.

A abordagem formal até agora discutida não contempla a variável tempo, que é de extrema importância na modelagem do comprtamento de sistemas físicos, tanto para análise quantitativa, como para análise qualitativa dos sistemas em tempo real.

2.2.5 Autômatos Temporizados Finitos

Para contemplar a variável tempo, (ALUR & DILL, 1994) propuseram uma modificação na teoria de autômatos finitos, incluindo o tempo, a qual foi denominada de teoria dos autômatos finitos temporizados. Neste contexto, o tempo foi modelado de forma discreta, e as seqüências de tempo evoluem de forma monotônicamente crescente de acordo com a seqüência dos seus índices.

O modelo de relógio fictício é similar ao modelo de tempo discreto, exceto que requer uma seqüência de tempos inteiros não decrescentes. A interpretação de uma execução temporizada deste modelo é que os eventos ocorrem em uma ordem específica em valores de tempo reais, mas somente é lido em tempos discretos.

Um autômato temporizado é um autômato finito com um conjunto finito de valores de tempo. Os relógios podem ser *resetados* para 0 (independentemente um do outro) com as transições dos autômatos, mantendo o rastreamento do tempo transcorrido desde o último *reset*.

Define-se a palavra temporizado como significando o comportamento de um sistema real sobre um alfabeto de eventos.

Uma seqüência $\tau = \tau_1, \tau_2, \cdots$ é uma seqüência infinita de valores de tempo $\tau_i \in \Re$ com $\tau_i > 0$, satisfazendo as seguintes restrições:

1. Monotonicidade: τ aumenta monotônicamente, isto é, $\tau_i < \tau_{i+1}$ para todo $i \geq 1$;
2. Progresso: Para todo $t \in \Re$, existe algum $i \geq 1$ tal que $\tau_i > t$;

Uma palavra temporizada sobre um alfabeto $\sum$ é um par *(σ,τ)* onde $\sigma = \sigma_1 \sigma_2 \cdots$ é uma palavra infinita sobre $\sum$, e τ é uma seqüência de tempo. Uma linguagem temporizada sobre $\sum$ é um conjunto de palavras temporizadas sobre $\sum$.

As operações sobre as linguagens, tais como interseção, união e concatenação são definidas e válidas para as linguagens temporizadas.

Quando um autômato realiza uma transição de estado, a escolha do próximo estado depende unicamente do símbolo de entrada lido. No caso de um autômato temporizado, esta escolha também depende do tempo relativo do símbolo de entrada aos tempos dos símbolos lidos previamente.

Um autômato temporizado é uma sextupla dada por $A = (\Sigma, Q, Q_0, Q_f C, E)$ (ALUR & DILL, 1994),(HERRMANN, 1998), onde:

- $\sum$ é um alfabeto finito;
- *Q* é um conjunto finito de estados;
- $Q_0 \subseteq Q$ é um conjunto finito de estados iniciais;
- $Q_f \subseteq Q$ é um conjunto finito de estados finais;
- C é um conjunto finito de relógios, e
- $E \subseteq Q \times Q \times [\Sigma \cup \{\Lambda\}] \times 2^C \times \Phi(C)$ relaciona um conjunto de transições. Um arco $(q, q', a, \lambda, \theta)$ representa uma transição do estado *q* para o estado *q'* quando da entrada do símbolo *a*. O conjunto $\lambda \subseteq C$ representa os relógios a serem resetados com esta transição, e θ é uma restrição de tempo sobre *C*.

Dado uma palavra temporizada (σ, τ) o autômato temporizado *A* inicia de um estado inicial no tempo 0 com todos os relógios inicializados em 0. Com o decorrer do tempo, os valores dos relógios mudam, refletindo o tempo transcorrido. No tempo τ_i, *A* muda do estado *q* para o estado *q'* de acordo com a transição da forma $(q.q', \sigma_i, \lambda, \theta)$ se tem-se como entrada o símbolo σ_i, e se o valor corrente do relógio

satisfaz θ. Com esta transição os relógios em λ são *resetados* para 0, e é iniciada a contagem do tempo com respeito ao tempo necessário para a nova ocorrência desta transição.

Um autômato temporizado é dito determinístico se:

1. tem somente um estado inicial, $|Q_0| = 1$;
2. para todo $q \in Q$, para todo $a \in \Sigma$, para todo par de arcos da forma $(q,-,a,-,\delta_1)$ e $(q,-,a,-,\delta_2)$, as restrições de tempo δ_1 e δ_2 são mutuamente exclusivas (isto é, $\delta_1 \wedge \delta_2$).

Exemplo: Considere um sistema simplificado mostrado na Figura 2.5. Há um local de armazenamento de materiais a serem reciclados (*la*), inicialmente com sua capacidade completa de 02 itens. Há um robô (*rb*) que transporta as peças desse local para uma máquina que processa peças (*mp*). Esse mesmo robô transporta a peça trabalhada na máquina para um local de consumo (*lc*). Sempre que uma peça é processada ou trabalhada, ela é transportada para o local de armazenamento (lc), sendo considerada boa para o consumo. Esse sistema apresenta 11 estados diferentes, os quais são descritos na Tabela 2.1.

Tabela 2.1: Descrição dos estados do sistema

Estados	Situação do Sistema
Estado inicial (0)	*la* com 2 peças, *lc* vazio, *mp* livre, *rb* livre;
Estado 1	*rb* com peça, *la* com 1 peça, *lc* vazio, *mp* livre;
Estado 2	*rb* livre, *la* com 1 peça, *lc* vazio, *mp* ocupada (não processando);
Estado 3	*rb* livre, *la* com 1 peça, *lc* vazio, *mp* ocupada (processando);
Estado 4	*rb* com peça, *la* com 1 peça, *lc* vazio, *mp* livre;
Estado 5	*rb* livre, *la* com 1 peça, *lc* com 1 peça, *mp* livre;
Estado 6	*rb* com peça, *la* vazio, *lc* com 1 peça, *mp* livre;
Estado 7	*rb* livre, *la* vazio, *lc* com 1 peça, *mp* ocupada (não processando);
Estado 8	*rb* livre, *la* vazio, *lc* com 1 peça, *mp* ocupada (processando);
Estado 9	*rb* com peça, *la* vazio, *mp* livre, *lc* com 1 peça;
Estado 10	*rb* livre, *la* vazio, *lc* com 2 peças, *mp* livre;

Definindo como estados iniciais: *lc* com pelo menos uma peça processada pronta para o consumo com sistema parado (*rb* e *mp* livres) e, também, toda peça processada já consumida e pronta para ser reciclada, tem-se que os estados iniciais

são os estados 0, 4 e 10. O alfabeto para este exemplo é dado pelos símbolos α, β, λ e μ, onde são definidos da seguinte forma:

- α - é o símbolo que representa o evento “rb pega peça”;
- β - é o símbolo que representa o evento “rb solta peça”;
- λ - é o símbolo que representa o evento “mp processa peça”;
- μ - é o símbolo que representa o evento “peça pronta”, isto é, peça já reciclada e boa para ser consumida.

O conjunto de relógios é definido pelos tempos das transições dado por δ_{ij} como sendo o tempo necessário para passar do estado *i* para o estado *j*, e de acordo com o processo e os estados acima descrito, a Figura 2.6 mostra o autômato temporizado que modela o sistema apresentado.

Figura 2.5: Sistema simplificado de reciclador de material com consumidor.

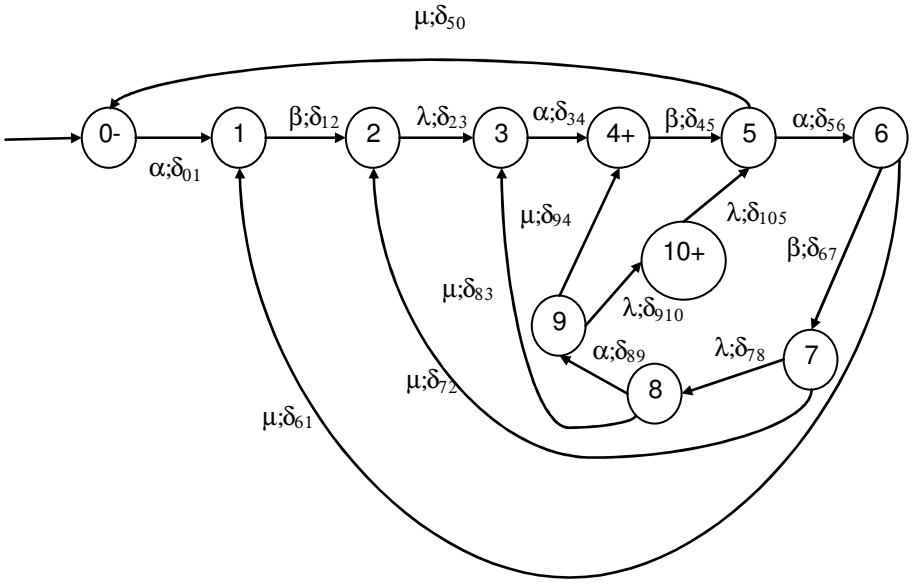

Figura 2.6: Modelo do sistema, representação de autômatos temporizados finitos.

2.2.5.1 Um Teorema de Kleene para os Autômatos Temporizados

O Teorema de Kleene citado anteriormente somente é válido para expressões regulares, obtidas a partir de símbolos através das operações de concatenação, união, e intersecção, e para os autômatos finitos, não sendo válido para os autômatos temporizados.

Entretanto Asarin (1997) usou o conceito dos autômatos temporizados (ALUR & DILL, 1994) com uma mudança no conceito de correspondência entre as linguagens a partir das seqüências temporizadas para os sinais de valores discretos e contínuos no tempo, definindo expressões regulares e *w*-regulares temporizadas.

Existem dois pontos diferenciais no uso do Teorema de Kleene original: primeiro, enquanto no teorema clássico a união é suficiente para a obtenção das expressões, para os autômatos temporizados, algumas vezes é necessário usar expressões com intersecção; segundo, quando traduzimos um autômato sobre um alfabeto $\sum$, pode-se criar uma expressão sobre um alfabeto maior $\sum'$ desde que a linguagem do autômato seja obtida de uma linguagem através de uma função remanescente $g : \Sigma' \to \Sigma$.

2.2.5.1.1 Expressões Regulares Temporizadas

Seja $\sum$ um alfabeto finito e, seja $\Re_+$ o conjunto dos números positivos reais. Um sinal sobre $\sum$ é uma função constante e contínua à esquerda dada por $\xi : (0,k] \to \Sigma$ para algum $k \in \Re_+ \cup \{0\}$ tal que ξ tem um número finito de descontinuidades. Todo sinal pode ser escrito como $\xi = a_1^{r_1} a_2^{r_2} ... a_n^{r_n}$ onde $a_1 \in \Sigma$, $r_i \in \Re_+$, $a_i \neq a_{i+1}$ e $\sum r_i = k$.

A concatenação de um conjunto de sinais é dada por $L_1 \circ L_2 = \{\xi_1 \circ \xi_2 : \xi_1 \in L_1 \wedge \xi_2 \in L_2\}$.

Definição (ASARIN, 1997): O conjunto $\Xi(\Sigma)$ de expressões regulares temporizadas sobre um alfabeto $\sum$, é definido recursivamente como parte de a, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, φ^* ou $<\varphi>_I$ onde $a \in \Sigma$, $\varphi, \varphi_1, \varphi_2 \in \Xi(\Sigma)$ e *I* é intervalo limitado e inteiro.

As semânticas das expressões regulares temporizadas $\| \ \| : \Xi(\Sigma) \to 2^{s(\Sigma)}$, são dadas por:

a. $\|a\| = \{a^r : r \in \Re_+\}$

b. $\|\varphi_1 \vee \varphi_2\| = \|\varphi_1\| \cup \|\varphi_2\|$

c. $\|\varphi_1 \wedge \varphi_2\| = \|\varphi_1\| \cap \|\varphi_2\|$

d. $\|\varphi_1 \cdot \varphi_2\| = \|\varphi_1\| \circ \|\varphi_2\|$

e. $\|\varphi *\| = \bigcup_{i=0}^{\infty} \left(\|\varphi^i\| \right)$

f. $\left\| \langle \varphi \rangle_I \right\| = \|\varphi\| \cap \{\xi : |\xi| \in I\}$

2.2.5.1.2 Transformação de Expressões Regulares Temporizadas em Autômatos

Asarin (1997) construiu os autômatos a partir de expressões de forma intuitiva. Estas formações intuitivas são mostradas nas figuras abaixo.

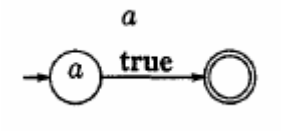

Figura 2.7a: O autômato para a pode mover-se a qualquer tempo do estado inicial para o estado final.

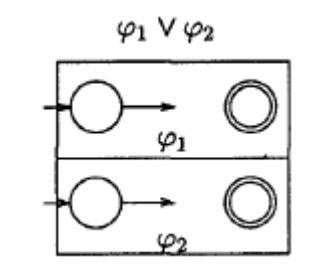

Figura 2.7b: Para a união de duas linguagens basta somente rodar o autômato em paralelo.

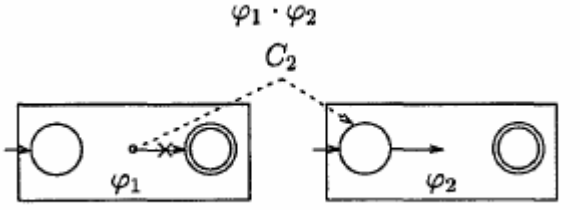

Figura 2.7c: Para concatenação, basta somente trocar qualquer transição para um estado final do primeiro autômato pela transição do estado inicial do segundo autômato (resetando seus relógios).

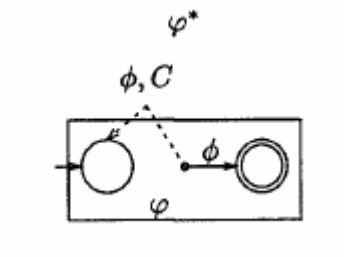

Figura 2.7d: Para a operação "*" basta adicionar transições do estado inicial para toda transição que leve ao evento requerido.

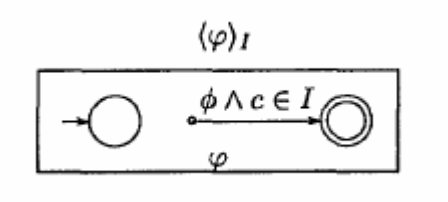

Figura 2.7e: Para a operação acima basta introduzir um novo relógio *c* e adicionar um teste para a guarda de toda a transição que leve ao evento requerido.

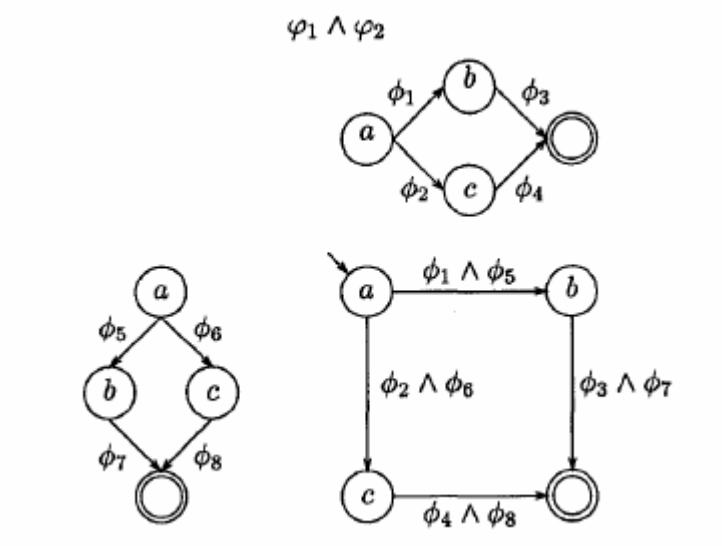

Figura 2.7f: Para a intersecção é necessário usar o produto cartesiano, levando-se em consideração que os símbolos (letras) estão associados aos estados e não às transições.

Formalmente as transformações de expressões para autômatos são dadas a seguir. Seja $A = (\Sigma, Q_1, Q_{01}, Q_{f1} C_1, E_1)$ e $A = (\Sigma, Q_2, Q_{02}, Q_{f2} C_2, E_2)$ autômatos temporizados aceitando as linguagens φ_1 e φ_2, respectivamente, então:

- O autômato para $\|a\|$ para todo $a \in \Sigma$ é $(\Sigma, \{q_1, q_2\}, 0, Q_f, \{q_1\}, \{q_2\})$;
- O autômato para $\|\varphi_1 \vee \varphi_2\|$ é:
  - $(\Sigma, Q_1 \cup Q_2, Q_{01} \cup Q_{02}, Q_{f1} \cup Q_{f2}, C_1 \cup C_2, E_1 \cup E_2)$
- O autômato para $\|\varphi_1 \wedge \varphi_2\|$ é:
  - $(\Sigma, Q, Q_0, Q_f, C_1 \cup C_2, E)$, onde:
    - $Q = \{\langle q_1, q_2 \rangle \in Q_1 \times Q_2, Q_{01}(q_1) = Q_{02}(q_2)\}$
    - $Q_0(\langle q_1, q_2 \rangle) = Q_{01}(q_1) = Q_{02}(q_2)$
    - $Q_f = Q \cap (Q_{f1} \times Q_{f2})$
    - $E = E_1 \times E_2$
- O autômato para $\|\varphi_1 \cdot \varphi_2\|$ é:
  - $(\Sigma, Q_1 \cup Q_2 - E_1, Q_{01} \cup Q_{02}, Q_{f1}, C_1 \cup C_2, E_2)$
- O autômato para $\|\varphi^*\|$ é:
  - $(\Sigma, Q_1, Q_{01}, Q_{f1} \cup E_1, C_1, E_1)$
- O autômato para $\|\langle \varphi \rangle_I\|$ é:
  - $(\Sigma, Q_1, Q_{01}, Q_{f1}, C_1 \cup \{c\}, E_1)$

A partir das equações acima se tem o Teorema de Kleene para os autômatos temporizados. E, da mesma forma para a transformação reversa, temos os seguintes Teoremas (ASARIN, 1997):

- Todas as linguagens temporizadas regulares podem ser aceitas por um autômato temporizado.
- A linguagem aceita por um autômato temporizado associada a um relógio livre de reset e, com uma condição inicial e um estado aceito pelo autômato é dita regular.
- A linguagem aceita por qualquer autômato temporizado associado a um relógio é dita regular.

- Toda linguagem aceita por autômato temporizado é uma renomeação de uma linguagem regular temporizada.

Corolário (ASARIN, 1997) – Teorema de Kleene para Autômatos Temporizados: Autômatos temporizados e expressões regulares temporizadas têm a mesma expressividade.

2.2.6 Redes de Petri

A Rede de Petri possui um grande potencial para a modelagem de sistemas com concorrência, e sistemas a eventos discretos. Este potencial é baseado em três aspectos fundamentais que compõem esta teoria: uma noção básica de não determinismo, uma noção básica de concorrência de processos e uma noção básica de seqüência (BEST, 90). Com estas três noções básicas é possível derivar uma grande variedade de outros conceitos e propriedades.

Como uma ferramenta gráfica, a Rede de Petri pode ser usada para a comunicação visual similarmente aos fluxogramas e diagramas de blocos. Adicionalmente, fichas (*tokens*) são usadas na Rede de Petri para simular a dinâmica e as atividades concorrentes dos sistemas. Como uma ferramenta matemática, é possível obter equações de estados, equações algébricas e outros modelos matemáticos para se estudar o comportamento dos sistemas. Rede de Petri pode ser usada por teóricos e práticos, ou seja, elas provêm uma boa forma de comunicação nestes dois ambientes: os práticos podem aprender com os teóricos a fazer modelos de sistemas mais metódicos, e os teóricos podem aprender com os práticos a fazer modelos de sistemas mais realistas (MURATA, 1989).

Historicamente falando, o conceito da Rede de Petri foi originalmente apresentado por Carl Adam Petri em sua tese de doutorado apresentada a faculdade de Matemática e Física da Universidade Técnica de Darmstadt (Alemanha) no ano de 1962.

Derivada da Teoria de Grafos e da representação dos autômatos finitos, a Rede de Petri também se aplica à otimização, análise e validação de sistemas, fornecendo, portanto, suporte às várias atividades essenciais no estudo de sistemas dinâmicos a eventos discretos. Concretamente, a modelagem e análise se aplicam ao comportamento dinâmico do sistema e à análise de suas propriedades

estruturais, a primeira que pode ser baseada em simulação e na resolução parcial da equação de estado e a segunda na derivação de propriedades para classes especiais de redes (DEL FOYO, 2001).

Na moderna literatura a Rede de Petri está divididas em três principais classes:

1. Redes de Petri Clássicas;
2. Redes de Petri Estendidas;
3. Redes de Petri de Alto Nível;

2.2.6.1 Redes de Petri Clássicas

Uma Rede de Petri é um grafo bipartido contendo dois tipos de nós (lugares e transições) e, que contém um estado inicial chamado de marcação inicial, M_0. Os arcos partem de um lugar para uma transição, ou de uma transição par um lugar. Na representação gráfica, os lugares são desenhados como círculos, e as transições como barras. Arcos orientados são identificados com seus pesos (inteiros positivos), onde um arco de peso k pode ser interpretado como um conjunto de k arcos em paralelo. A identificação para o peso unitário é usualmente omitido. A marcação (estado) relacionada a cada lugar é definido por um número inteiro não negativo de fichas neste lugar.

Formalmente uma Rede de Petri é uma quíntupla $PN = (P, T, F, W, M_0)$ (MURATA, 1989) onde:

- $P = \{p_1, p_2, \cdots, p_m\}$ é um conjunto finito de lugares;
- $T = \{t_1, t_2, \cdots, t_n\}$ é um conjunto finito de transições;
- $F \subseteq (P \times T) \cup (T \times P)$ é um conjunto de arcos orientados;
- $W : F \to \{1,2,3,\cdots\}$ é a função peso;
- $M_0 : P \to \{1,2,3,\cdots\}$ é a marcação inicial;
- $P \cap T = \phi$ e $P \cup T \neq \phi$

Uma estrutura de Rede de Petri sem qualquer especificação de marcação inicial é dada por N, onde $N = (P, T, F, W)$. Uma Rede de Petri com a marcação inicial definida é dada pelo par (N, M_0).

De modo a simular o comportamento dinâmico de um sistema, os estados ou marcações de uma Rede de Petri são modificados de acordo com a seguinte regra de transição ou disparo:

1. Uma transição *t* é dita habilitada se todos os lugares de entrada *p* de *t* estão marcados com no mínimo *w(p,t)* fichas, onde *w(p,t)* é o peso dos respectivos arcos de *p* para *t*.
2. Uma transição habilitada pode ou não disparar;
3. Um disparo de uma transição habilitada t remove *w(p,t)* fichas de cada lugar de entrada *p* de *t*, e adiciona *w(t,p)* fichas para cada lugar de saída *p* de *t*, onde *w(t,p)* é o peso do arco de *t* para *p*.

Um par de arcos entre um lugar *p* e uma transição *t* pode ser chamado de laço, se *p* é duplamente um lugar de entrada e um lugar de saída de *t*. Uma Rede de Petri é dita pura se não contém laços. Uma Rede de Petri é dita ordinária se todos os pesos dos arcos são iguais a 1.

Uma Rede de Petri (N, M_0) é dita limitada (ou de capacidade finita) se nenhum lugar *p* não ultrapassa um número máximo de fichas, isto é, se $M(p) \leq k$, para todo *p*. Neste caso, para uma transição *t* ser habilitada, é necessário também que o número de fichas de cada lugar de saída *p* de *t* não exceda a capacidade *K(p)* após o disparo de *t*. Esta regra de transição é chamada de regra de transição estrita.

2.2.6.2 Redes de Petri Estendidas

As Redes de Petri Estendidas, por outro lado, correspondem a modelos para os quais as regras de transição sofrem algumas variações com a finalidade de aumentar a capacidade de representação do modelo. Aqui se podem considerar três tipos de subclasses:

- As que têm o poder de representação de máquinas Turing (Redes de Petri com arcos inibidores);
- As que permitem a modelagem de Redes de Petri Híbridas e Redes de Petri contínuas;
- As que correspondem a modelos que descrevem o funcionamento de sistemas cuja evolução vai depender de eventos externos e/ou da

variável tempo (Redes de Petri sincronizadas, Redes de Petri temporizadas e Redes de Petri estocásticas);

2.2.6.3 Redes de Petri de Alto Nível

As Redes de Petri clássicas e estendidas permitem a modelagem de estados, eventos, condições, sincronização de processo, paralelismo de processo, escolha e interação de processo. Entretanto, existem casos onde se deseja descrever processos reais que tendem a ser mais complexos e abrangentes, onde as Redes de Petri Clássicas não permitem a modelagem dos dados. Para resolver este problema algumas extensões têm sido propostas:

a) Extensão para modelagem dos dados:

- Nas Redes de Petri Coloridas (JENSEN, 1996) as fichas possuem um valor, denominado “cor”. As transições determinam os valores das fichas produzidas baseada nos valores das fichas consumidas, isto é, as transições estabelecem o relacionamento entre os valores das fichas de entrada e os valores das fichas de saída, podendo ser também especificadas pré-condições de cores para as fichas a serem consumidas.

b) Extensão com hierarquia para a representação e estruturação de modelos ou sistemas de grande porte:

- Para se obter especificações mais precisas dos sistemas reais, é considerada uma extensão onde se incorpora os conceitos de hierarquia e sub-redes. Uma sub-rede é um agregado de um número de lugares, transições, e sub-sistemas. Desta maneira uma construção (sub-rede) pode ser usada para estruturar uma variedade de processos. Se em um nível tem-se uma descrição macro do processo (sem ter que considerar todos os detalhes), em outro nível tem-se o comportamento mais detalhado. Assim a abordagem através de hierarquia nos permite esta abordagem. As Redes GHENeSys, Coloridas e ML são exemplos desta aplicação.

Estas extensões das Redes de Petri (que possuem cor e hierarquia) são denominadas Redes de Petri de Alto Nível.

2.2.6.4 Propriedades da Rede de Petri

Existem dois tipos de propriedades que podem ser estudadas pelas Redes de Petri: as propriedades comportamentais (que dependem da marcação inicial) e as propriedades estruturais (não dependem da marcação inicial).

As propriedades comportamentais mais comumente usadas são:

- Alcançabilidade: uma marcação M_n é dita alcançável a partir de M_0 se existe uma seqüência de disparos que transforma M_0 em M_n. Neste caso, M_n é alcançável a partir de M_0 pela seqüência de disparos σ e, denotamos $M_0 [\sigma > M_n$.
- Limitação: uma Rede de Petri é dita k-limitada ou simplesmente limitada se o número de fichas em cada lugar nunca excede um número finito *k* para qualquer marcação alcançável a partir de M_0. Uma Rede de Petri (N, M_0) é dita segura se é 1-limitada.
- Vivacidade: uma Rede de Petri é dita viva se apresenta a ausência completa de bloqueios na operação do sistema. Ou seja, uma rede (N, M_0) é dita viva se, independentemente da marcação alcançável a partir de M_0, qualquer transição da rede é possível ser disparada partindo desta marcação através de uma determinada seqüência de disparos. Porém, esta propriedade é muito forte para ser verificada. Por este motivo em (MURATA, 1989) se define diferentes níveis de vivacidade. Assim dada uma Rede de Petri (N, M_0), uma transição $t \in T$ é dita:
  - i. Morta (L0-viva), se *t* não aparece em nenhuma seqüência de disparo de L(M_0);
  - ii. Potencialmente disparável (L1-viva), se *t* aparece ao menos uma vez em alguma seqüência de disparo de L(M_0).
  - iii. L2-viva, se, dado um número $k \in \mathbb{N}^+$, $k > 1$, *t* aparece ao menos *k* vezes em alguma seqüência de disparo de L(M_0).
  - iv. L3-viva, se *t* aparece infinitas vezes em alguma seqüência de disparo de L(M_0).
  - v. L4-viva ou simplesmente viva se *t* é L1 viva para cada marcação M em R(M_0). Este é o nível de vivacidade mais

forte e corresponde ao conceito de vivacidade expresso inicialmente.

- Reversibilidade: uma Rede de Petri (N, M_0) é dita reversível se, para cada marcação *M* em *R(M0)*, *M0* é alcançável a partir de *M*.
- Distância Síncrona: define-se a distância síncrona entre duas transições t_1 e t_2 de uma Rede de Petri (N, M_0) por $d_{12} = \max\left|\bar{\sigma}(t_1) - \bar{\sigma}(t_2)\right|$, onde σ é uma seqüência de disparo partindo de uma marcação qualquer *M* em *R(M0)* e $\bar{\sigma}(t_1)$ é o número de vezes que a transição t_i dispara em σ.

São ditas propriedades estruturais das Redes de Petri, as propriedades que não dependem da marcação inicial mas somente da estrutura topológica da rede. As propriedades estruturais são:

- Vivacidade Estrutural: uma Rede de Petri N é dita estruturalmente viva se existe uma marcação inicial viva para N.
- Controlabilidade: uma Rede de Petri é dita completamente controlável se qualquer marcação é atingível a partir de uma dada marcação.
- Limitação Estrutural: uma de Rede de Petri N é dita limitada estruturalmente se é limitada para toda marcação inicial *M0*.
- Conservabilidade: uma Rede de Petri N é dita parcialmente conservativa, se existe um inteiro positivo y(p) para algum lugar p, tal que a soma ponderada de fichas (tokens) seja constante, isto é, $M^T y = M_0{}^T y = cte$; de forma análoga, uma Rede de Petri N é dita totalmente conservativa, se existe um inteiro positivo y(p) para cada lugar p, tal que a soma ponderada de fichas (tokens) seja constante, isto é, $M^T y = M_0{}^T y = cte$.
- Consistência: uma Rede de petri é dita parcialmente consistente se existe uma marcação inicial M_0 e uma seqüência de disparos σ que leva ciclicamente a M_0 de modo que (alguma) cada transição ocorre pelo menos uma vez em σ.

O comportamento dinâmico de grande parte dos sistemas estudados pela engenharia podem ser descritos usando equações diferenciais. No caso de sistemas

modelados por Rede de Petri este comportamento dinâmico também é regido por sua equação de estado. Na próxima seção é apresentada a representação algébrica da Rede de Petri, ou seja, a equação de estado.

2.2.6.5 Equação de Estado

Uma parte fundamental da equação de estado é o que se denomina como matriz de incidência (MURATA, 1989). A matriz de incidência representa a estrutura de uma rede. Para uma Rede de Petri *N* com *n* transições e *m* lugares, a matriz de incidência $A = [a_{ij}]$ é uma matriz *n x m* dada por $A = A^+ - A^-$.

A matriz A^+ de ordem *n x m* representa a quantidade de fichas adicionadas aos lugares após o disparo de uma transição, sendo que, $A^+ = [a_{ij}^{\ +}]$, onde, $a_{ij}^{\ +} = w(i, j)$ é o peso do arco que leva da transição *i* ao lugar *j*.

A matriz A^- de ordem *n x m* representa a quantidade de fichas retiradas dos lugares após o disparo de uma transição, sendo que, $A^- = [a_{ij}^{\ -}]$, onde, $a_{ij}^{\ -} = w(j,i)$ é o peso do arco que leva do lugar *j* a transição *i*.

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

$$M_{k+1} = M_k + A^t v_k \tag{13}$$

onde:

- M_{k+1} e M_k são respectivamente vetores colunas de *m x 1*, que representam as marcações sucessora e atual, respectivamente.
- v_k é o vetor coluna de *n x 1* que representa o vetor de habilitação ou disparo.

Se considerarmos que uma marcação M_d é alcançável a partir de M_0 através da seqüência de disparo $\{v_1, v_2, \cdots, v_d\}$ pode-se escrever a equação de estado para:

$$M_d = M_0 + A^t \sum_{k=1}^{d} v_k \tag{14}$$

que pode ser reescrita da seguinte forma:

$$A^t x = \Delta M \tag{15}$$

onde $\Delta M = M_d - M_0$ e $x = \sum_{k=1}^{d} v_k$

O i-ésimo componente do vetor *x* denota a quantidade de vezes que a i-ésima transição deve ser disparada até alcançar a marcação M_d.

Existem outras propriedades nas redes que podem ser calculadas a partir da matriz de incidência. A solução da equação homogênea $Ay = 0$ é chamada de invariantes de lugar ou *p*-invariantes. Da mesma forma, a solução da equação homogênea $A^t x = 0$ é chamada de invariantes de transição ou *t*-invariantes. As soluções p-invariantes e t-invariantes são usadas na análise estrutural das Redes de Petri para a verificação das propriedades: limite estrutural, conservatividade, repetividade e consistência.

2.2.6.6 Refinamento na Rede de Petri

A técnica de refinamento consiste em substituir uma transição ou um lugar por uma rede. Um conceito relacionado com o refinamento é a redução que consiste em substituir um subconjunto de elementos por um lugar ou uma transição, (MURATA, 1989) apresenta um conjunto de regras de redução que preservam propriedades como vivacidade, segurança e limitação. A Figura 2.8 ilustra estas transformações.

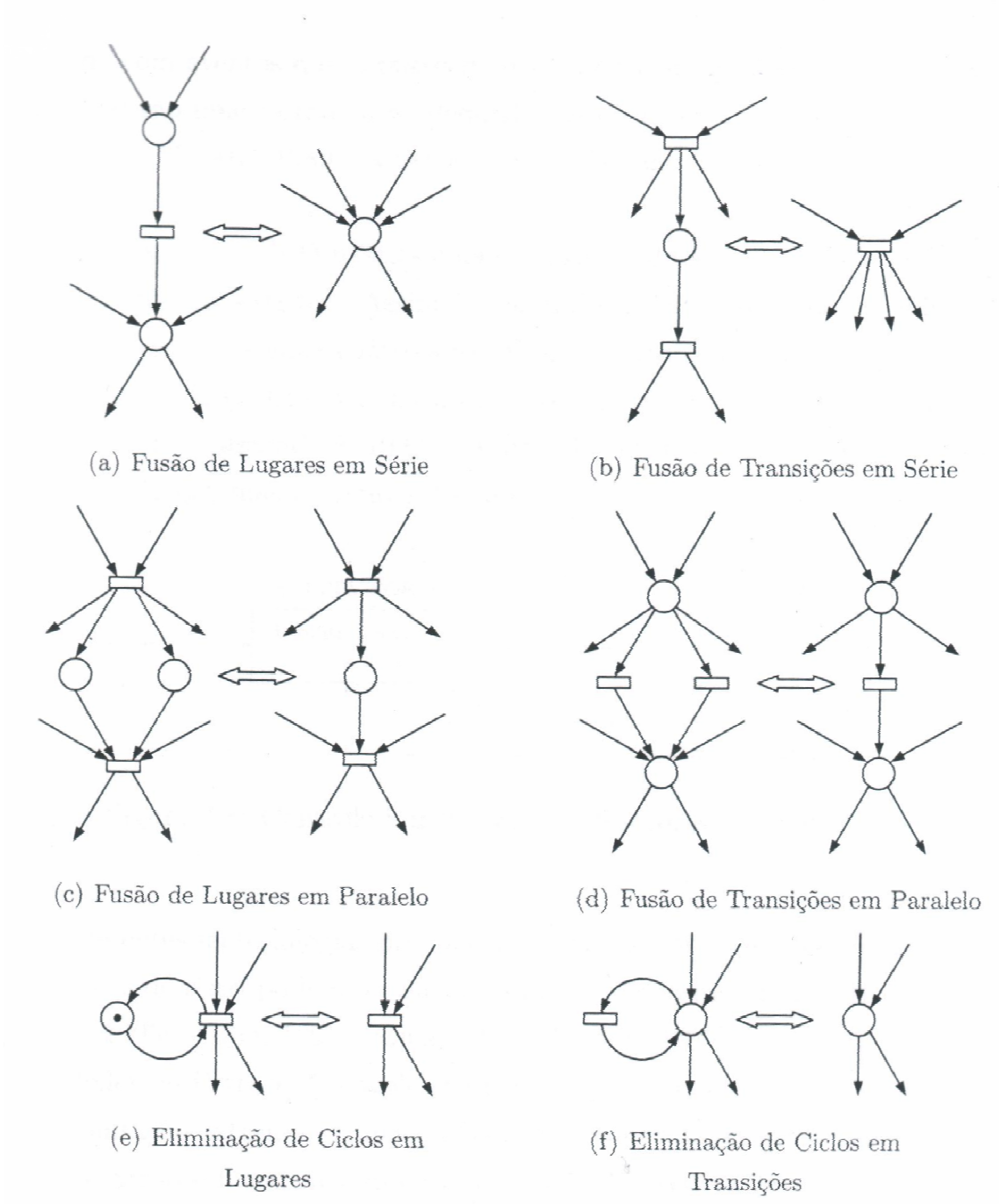

Figura 2.8: Transformações preservando vivacidade, segurança e limitação.

2.2.6.7 Rede de Petri a partir de Grafo de Transição

Pelas transições de palavras com símbolos a partir de um dado alfabeto, as transições podem ser interpretadas como uma ocorrência de eventos ou tarefas de execução dentro de um sistema. Rede de Petri é usada em uma variedade de

aplicações, sendo a análise de performance e verificação de tempo algumas delas. Uma Rede de Petri pode ser derivada de qualquer modelo de sistema especificado em grafos de transição constituído por símbolos obtidos a partir de um alfabeto de eventos. Este método é baseado na teoria das regiões para Sistema de Transição Elementares (STE). Para qualquer STE, existe uma Rede de Petri com um mínimo de transições realizáveis (uma transição para cada palavra) em que a árvore de alcançabilidade é isomórfica ao grafo de transição original. As Redes de Petri obtidas a partir deste método de síntese são seguras e, podem derivar diferentes tipos de redes (ex.: pura, livre escolha) a partir de um grafo de transição a depender das restrições impostas (CORTADELLA, 1998).

Uma Rede de Petri expressa o mesmo comportamento de um grafo de transição, conforme mostrado na Figura 2.9 abaixo:

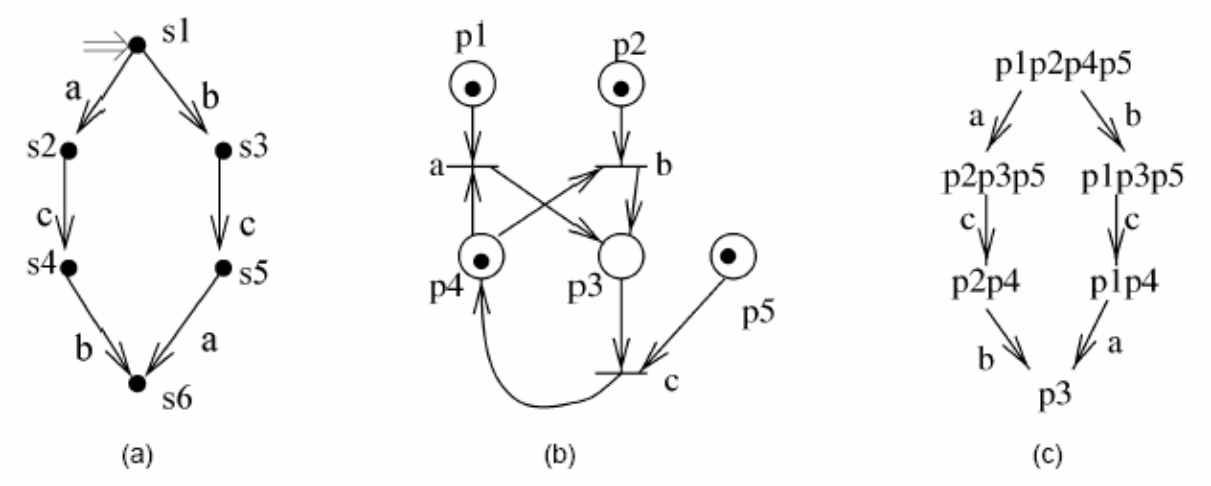

Figura 2.9: Um exemplo de grafo de transição (a); a Rede de Petri correspondente (b), e sua árvore de alcançabilidade isomórfica ao grafo de trans. (c).

Na estrutura da Rede de Petri existe uma correspondência de toda a transição da rede com um símbolo a partir do alfabeto de entrada. Para a síntese da Rede de Petri a partir de um grafo de transição, com árvore de alcançabilidade isomórfica ao grafo de transição, (CORTADELLA, 1998) propôs o seguinte algoritmo:

- Para cada evento $e \in E$, gera uma transição nomeada de e na Rede de Petri;
- Para cada região $r_i \in R_{GT}$, gere um lugar r_i na Rede de Petri. Onde, intuitivamente podemos dizer que uma região é o conjunto ou subconjunto dos estados alcançáveis do grafo de transição. Assim, existe uma correspondência bi-unívoca dos estados de uma região e

as marcações de uma Rede de Petri nos quais estes lugares possuem uma ficha.

- Um lugar r_i contém uma ficha na marcação inicial se a correspondente região de r_i contém o estado inicial do grafo de transição.

2.2.7 Rede de Petri Temporizadas

A Rede de Petri Temporizada é um tipo de rede estendida que introduz novos elementos e abordagens sem alterar a teoria das redes, contribuindo para aumentar o poder de representação das redes sem perder o poder de análise.

Para a análise de performance e planejamento de sistemas dinâmicos, é extremamente necessária a introdução de tempos de atraso associados com as transições ou lugares nos modelos destas redes. Estes tempos podem ser determinísticos, definindo a Rede de Petri Temporizada, ou, tais tempos pode ser uma função de modelos probabilísticos, definindo a Rede de Petri Estocástica.

Os tempos podem estar associados aos lugares ou às transições, sendo as redes chamadas de Redes de Petri p-temporizadas ou t-temporizadas, respectivamente.

Existem dois modelos básicos de Redes de Petri t-temporizadas que vêm sendo desenvolvidas nos últimos tempos (BERTHOMIEU & DIAZ, 1991):

- Modelo de Ramchandani
- Modelo de Merlin

O modelo de Rede de Petri derivado de Ramchandani vem da associação de uma duração finita de disparo para cada arco orientado de transição da rede. A regra de disparo clássica das Redes de Petri foi modificada para “contar” o tempo que leva até o disparo de uma transição e, para expressar que a transição deve disparar tão logo esteja habilitada. Este modelo de rede tem sido usado principalmente para a avaliação de performance.

O modelo de Rede de Petri derivado de Merlin é mais geral que o modelo de Ramchandani. O modelo definido por Merlin utiliza dois valores de tempo, dois números reais, a e b, com $a \leq b$, que são associados com cada transição t_i:

i. *a* ($0 \leq a$), é o mínimo tempo que deve decorrer, a partir do instante no qual a transição t_i foi habilitada, até que esta transição possa ser disparada;

ii.b ($0 \leq b \leq \infty$) denota o tempo máximo de duração na qual a transição t_i pode estar habilitada sem ter sido disparada.

Os tempos *a* e *b*, para a transição t_i, são relativos ao momento na qual a transição t_i está habilitada. Assumindo que uma transição t_i foi habilitada no tempo τ, então t_i, se continuamente habilitada, não pode disparar antes do tempo $\tau + a$ e deve disparar antes ou no tempo $\tau + b$, a menos que seja desabilitada antes do disparo pelo disparo de alguma outra transição concorrente.

Formalmente uma Rede de Petri Temporizada é uma sextúpla $PN = (P, T, F, W, M_0, \delta)$ (BERTHOMIEU & DIAZ, 1991) onde:

- $P = \{p_1, p_2, \cdots, p_m\}$ é um conjunto finito de lugares;
- $T = \{t_1, t_2, \cdots, t_n\}$ é um conjunto finito de transições;
- $F \subseteq (P \times T) \cup (T \times P)$ é um conjunto de arcos;
- $W : F \rightarrow \{1,2,3,\cdots\}$ é a função peso;
- $M_0 : P \rightarrow \{0,1,2,3,\cdots\}$ é a marcação inicial;
- $\delta : T \rightarrow Q^+ \times (Q^+ \cup \infty)$, onde Q^+ é o conjunto dos números racionais positivos;
- $P \cap T = \phi$ e $P \cup T \neq \phi$

Desta forma no modelo de Ramchandani cada transição t_i tem um tempo δ associado. Já de acordo com o modelo de Merlin temos que cada transição está associada a um par $\delta = (a, b)$, onde *a* é denominado o tempo mais cedo de disparo (ou EFT, do inglês *earliest firing time*) e *b* é denominado o tempo mais tarde de disparo (ou LFT, do inglês *latest firing time*).

Quando uma transição temporizada é habilitada e é transcorrido o tempo de duração correspondente (diaparo), as fichas do lugar correspondente são removidas das pré-condições e adicionadas às pós-condições associadas e, os relógios associados são resetados. Na ocorrência de uma transição temporizada ficar desabilitada antes da ocorrência do disparo, o relógio correspondente deve ser também resetado.

A regra de disparo é formalmente expressa pelas seguintes condições:

1. Deve satisfazer a todas as condições de disparo de uma Rede de Petri;
2. Expressa o fato que a transição habilitada não pode disparar antes de EFT e deve disparar antes ou em LFT a menos que outros disparos ocorram modificando os estados atuais.

Assim, a transição t_i é disparável a partir de um estado *S* no tempo $\tau + \theta$ se as seguintes condições são satisfeitas:

a) t_i está habilitada pela marcação M no tempo τ;

b) O tempo relativo de disparo θ, relativo ao tempo de disparo absoluto τ, não é menor que EFT da transição t_i e não é maior que os LFTs de todas as transições habilitadas pela marcação M, isto é, EFT de $t_i \leq \theta \leq$ min{LFT de t_k }, onde *k* é a faixa do conjunto de transições habilitadas por M.

O tempo de atraso θ não é um tempo global, e pode ser visto como um relógio local e virtual para cada transição, e que deve estar na mesma escala de tempo para todas as transições na Rede de Petri Temporizada.

O disparo de uma transição não pode ser interrompido e, também não se deve permitir uma transição iniciar um segundo disparo antes do primeiro estar finalizado.

A presença de conflitos estruturais (uma estrutura que possui um lugar com duas ou mais transições de saída) em uma Rede de Petri, necessita de um mecanismo de execução que soluciona este conflito (somente uma transição pode disparar, as demais transições devem ficar desabilitadas). Caso este mecanismo seja baseado sob uma função de probabilidade, a rede passa a ser denominada estocástica. Por esta razão, o uso da Rede de Petri Temporizada Determinística para a análise de performance tem sido restrito para a rede livre-escolha ou livre de conflito, a qual pode ser modelada como grafo marcado, ou grafo de transição (ZURAWSKI & ZHOU, 1994).

2.2.7.1 Equação de Estado para Redes de Petri Temporizada

Baseado na equação de estado definida em (13), temos a seguinte equação de estado para as Redes de Petri Temporizadas:

$$M_{k+1} = M_k + A^t v_{kt} \tag{16}$$

onde:

- M_{k+1} e M_k são respectivamente vetores colunas *m x 1*, que representam as marcações sucessora e atual, respectivamente.
- v_{kt} é o vetor coluna de *n x 1* que representa o vetor de habilitação ou disparo correspondente à regra de disparo da Rede de Petri Temporizada.

Assim,

$$v_{kt} = v_k, \; se \; a_n \leq \theta_n \leq b_n \tag{17}$$

Isto é,

$$v_{kt}(n) = \begin{cases} 1, & ou \; v_k(n), \; se \; a \leq \theta \leq b \\ 0, & em \; caso \; contrario \end{cases} \tag{18}$$

2.2.7.2 Propriedades das Redes de Petri Temporizada

Classes de estado é conjunto de todos os estados alcançáveis a partir do estado inicial através de uma seqüência de disparo ω.

O conjunto de marcações de uma Rede de Petri Temporizada que pode ser alcançável a partir de um estado inicial M_0 será denotado por R(M_0).

O problema alcançabilidade é provar se uma dada marcação pertence ou não a R(M_0).

O problema de limitação é saber se todas as marcações em R(M_0) são limitadas ou não, isto é, $\exists k \in \aleph, \forall M \in R(M_0), \forall p \in P \mid M(p) \leq k$.

Uma Rede de Petri Temporizada é dita T-limitada se existe um número natural positivo *k* em que nenhuma das transições pode ser habilitada mais do *k* vezes simultaneamente por qualquer marcação alcançável.

A propriedade T-viva é um caso particular da propriedade T-limitada com *k = 1*.

De acordo com Berthomieu & Diaz (1991) e Starke (1990), as seguintes propriedades são válidas:

A. Indeterminabilidade
    i. Teorema da Indeterminabilidade: Os problemas de alcançabilidade e limitação são indetermináveis;

B. Limitação

i. Lema 01: As constantes α_i, β_i e γ_{jk} de qualquer domínio computado a partir do estado inicial pela regra de disparo são combinações lineares com coeficientes inteiros dos EFTs e LFTs associados com as transições da Rede de Petri Temporizada, isto é:

   a. $\forall i \exists \lambda_1, \cdots, \lambda_{2n} \in Q$

   b. $\alpha_i = \lambda_1 \alpha_1 + \cdots + \lambda_n \alpha_n + \lambda_{n+1} \beta_1 + \cdots + \lambda_{2n} \beta_n$

   c. e, similarmente, para cada β_i e γ_{jk}

   d. onde, α_i é o menor valor possível da variável *t(i)*; β_i é o maior valor possível da variável *t(i)*; γ_{jk} é o maior valor possível da diferença *t(j) – t(k)*.

ii. Lema 02: Seja *A* e *B* duas constantes limitadas e seja $q_1, \cdots, q_n$ um conjunto finito de constantes racionais. Existe somente um número limitado de combinações lineares dos números $q_1, \cdots, q_n$ com coeficientes inteiros, entre os números *A* e *B*, isto é, o número X dos números racionais tal que $X = \lambda_1 q_1 + \cdots + \lambda_n q_n$, e $\lambda_1, \cdots, \lambda_n \in Q$, e $q_1, \cdots, q_n \in Q$, e $A \leq X \leq B$ é limitado.

iii. Lema 03: Se uma Rede de Petri Temporizada é T-limitada, então o conjunto de todos os domínios de disparos *D*, e as classes de estados dele é finito.

iv. Teorema 02: Uma Rede de Petri Temporizada tem um número de classes de estados limitadas se e somente se é limitada.

v. Teorema 03: Uma Rede de Petri T-limitada é limitada se nenhum par de classes de estado *C=(M,D)* e *C'=(M',D')* são alcançáveis a partir da classe de estado inicial, tal que *C'* é alcançável a partir de *C*, e *M'(p)≥M(p).*

vi. Uma Rede de Petri Temporizada é limitada se nenhum par de classes de estado *C = (M,D)* e *C' = (M', D')*, alcançável a

partir das classes de estado inicial, satisfaz plenamente às seguintes condições:

a. *C'* é alcançável a partir de *C*;
b. *M'(P)* ≥ *M(p)*;
c. *D' = D*;
d. $\forall p \in \{p \in P \mid M'(p) > M(P)\}$ então $\left(M(p) > \max A^{-}(t_i, p)\right)$.

2.2.7.3 Relação entre Rede de Petri e os Autômatos

Autômatos com relações de concorrência são grafos de transição com uma coleção de estados dependentes de relações distintas para as ações. (MANFRED, 2002) mostra e propõe como associar cada Rede de Petri (lugar/transição) com um autômato com mesmo comportamento dinâmico.

Para cada Rede de Petri $PN = (P, T, F, M_0)$ podemos associar um autômato $A = (Q; \Sigma; \delta; q_0; Q_m)$, onde:

1. O espaço de estados *M* do autômato *A* é o conjunto de todas as marcações da Rede de Petri *PN*;
2. O conjunto de eventos *E* é o mesmo apresentado na Rede de Petri *PN*;
3. O conjunto de transições T é o mesmo;
4. O estado inicial é a marcação inicial M_0;

2.2.7.4 Equivalência entre Autômato Temporizado e Rede de Petri Temporizada

Em Haar (2002) é demonstrado que o autômato temporizado é equivalente à Rede de Petri Temporizada derivada de Merlin, através de algoritmos que demonstram a relação entre os autômato temporizado e a Rede de Petri Temporizada. Cada classe de estado corresponde à marcação *M* e o domínio *D* de disparos, que coleta os valores de relógios possíveis se *M* é alcançável a partir de uma particular seqüência de disparos (não temporizada). Desde que *M* possa ser alcançada por duas ou mais seqüências diferentes, o grafo das classes obtido é em

geral maior que a árvore de alcançabilidade. Este método possibilita que a árvore de alcançabilidade seja isomorfa ao grafo do autômato temporizado.

2.2.8 Ferramentas de Simulação Para Redes Temporizadas

Para a escolha da ferramenta de simulação, utilizaram-se como referência os *softwares* listados no site do ***DAIMI*** (*Departament of Computer Science, University of Aarhus*), considerando-o como representativo da área para o levantamento das características gerais dos sistemas e ambientes de modelagem em Redes de Petri Temporizadas existentes. As principais características dos 21 sistemas que tratam de Redes de Petri Temporizadas são apresentadas na Tabela 2.2.

Para a escolha do software a ser utilizado nas simulações foram utilizados os seguintes critérios:

1. Características: aspecto avaliado em três notas (1,3 e 5), onde a nota máxima representa a situação onde há a disponibilidade de um editor gráfico, jogador de marcas, recursos para análise de performance, espaço de estados, algoritmos para o cálculo de invariantes de lugar e de transição; a nota intermediária é dada quando ao menos duas destas funcionalidades estão ausentes; a nota mínima é dada quando mais de duas destas funcionalidades estão ausentes.
2. Sistema Operacional: aspecto avaliado em três notas (1, 3 e 5), onde a nota máxima é dada para a compatibilidade com o sistema Windows; a nota intermediária é dada para a compatibilidade com o sistema Linux; e a nota mínima é dada para a compatibilidade do sistema com DOS ou Sun.
3. Licença: aspecto avaliado em duas notas, 5 para distribuição gratuita, e 3 para distribuição em caráter comercial;

A Tabela 2.3 apresenta a pontuação obtida para cada um dos 21 softwares. As ferramentas JFern, PetriNet Toolbox, PNTalk, Renew, Tina e Visual Object Net empataram na avaliação. Dentre estas ferramentas foi escolhida a ferramenta **Tina**, entre as gratuitas, por termos um melhor canal de comunicação com os desenvolvedores do sistema. Entretanto, a melhor ferramenta é PetriNet Toolbox (MATLAB), pois apesar de ser comercial, é a única que atualmente possibilita a análise de performance, objetivo principal do trabalho.

Tabela 2.2: Ferramentas de Simulação disponíveis para Redes de Petri Temporizadas

No.	Nome do Software	Características	Sist. Operacional	Ambiente	Licença	Atualização
1	JFern	Editor Gráfico Animação com Jogador de Marcas Simulação Rápida Espaço de Estados Análise de Desempenho Simples Intercâmbio de Formato de Arquivo		Java	Gratuita	Maio/2004
2	JPetriNet	Editor Gráfico Análise Estrutural		Java	Gratuita	Março/2004
3	MISS-RdP	Editor Gráfico Simulação Rápida	Sun, SunOS PC, MS Windows NT PC, MS Windows 2000	C++	Comercial	Maio/2004
4	Opera	Editor Gráfico Animação com Jogador de Marcas Simulação Rápida Espaço de Estados Espaço de Estados Condensado Invariantes de Lugar Invariantes de Transição Reduções da Rede Análise Estrutural Análise de Desempenho Simples Análise de Desempenho	MS DOS		Comercial	Maio/2004

No.	Nome do Software	Características	Sist. Operacional	Ambiente	Licença	Atualização
		Avançada Intercâmbio de Formato de Arquivo				
5	PACE	Editor Gráfico Animação com Jogador de Marcas Simulação Rápida Reduções da Rede Técnicas *Fuzzi*, Otimizações da Rede	PC, MS Windows 95 PC, MS Windows 98 PC, MS Windows NT PC, MS Windows 2000 PC, MS Windows XP		Comercial	Marco/2003
6	PED	Editor Gráfico	Sun Linux		Gratuita	Maio/2003
7	PEP	Editor Gráfico Animação com Jogador de Marcas Espaço de Estados Espaço de Estados Condensado Invariantes de Lugar Invariantes de Transição Reduções da Rede Análise Estrutural Intercâmbio de Formato de Arquivo Checagem do Modelo Geradores de Redes de Petri	Sun Linux		Gratuita	Setembro/2004

No.	Nome do Software	Características	Sist. Operacional	Ambiente	Licença	Atualização
8	Petri .NET Simulator	Editor Gráfico Animação com Jogador de Marcas Simulação Rápida Análise de Desempenho Simples Blocos de Subsistema	PC, MS Windows 98 PC, MS Windows NT PC, MS Windows 2000 PC, MS Windows XP		Comercial	Agosto/2004
9	Petri Net Toolbox	Editor Gráfico Animação com Jogador de Marcas Simulação Rápida Espaço de Estados Invariantes de Lugar Invariantes de Transição Análise Estrutural Análise de Desempenho Simples Análise de Desempenho Avançada Intercâmbio de Formato de Arquivo Análise Max-plus para Marked Graphs		MATLAB 6.0 mínimo	Comercial	Janeiro/2004
10	PetriSim	Editor Gráfico Simulação Rápida	MS DOS		Gratuita	Maio/2003
11	PNtalk	Editor Gráfico Animação com Jogador de Marcas Simulação Rápida	Sun MS Windows		Gratuita	Maio/2003

No.	Nome do Software	Características	Sist. Operacional	Ambiente	Licença	Atualização
		Análise de Desempenho Simples				
12	PROTOS	Editor Gráfico Simulação Rápida Análise Estrutural Análise de Desempenho Simples Rápida Prototipagem Sistema de Gerenciamento do Fluxo de Trabalho	PC, MS Windows 98 PC, MS Windows NT PC, MS Windows 2000 PC, MS Windows XP		Comercial	Junho/2004
13	Renew	Editor Gráfico Animação com Jogador de Marcas Simulação Rápida Intercâmbio de Formato de Arquivo Rápida Prototipagem Sistema de Gerenciamento do Fluxo de Trabalho		Java	Gratuita	Maio/2003
14	Romeo	Editor Gráfico Espaço de Estados	PC, Linux Macintosh, Mac OS X	Tcl/Tk	Gratuita	Maio/2003
15	SEA	Editor Gráfico Animação com Jogador de Marcas Simulação Rápida Visualização Gráfica Abstrata	Sun		Gratuita	Maio/2003
16	Simulaworks	Editor Gráfico Simulação Rápida	PC, MS Windows 2000 PC, MS Windows XP		Comercial	Agosto/2004

No.	Nome do Software	Características	Sist. Operacional	Ambiente	Licença	Atualização
17	StpnPlay	Editor Gráfico Simulação Rápida Análise de Desempenho Simples Intercâmbio de Formato de Arquivo	PC, MS Windows 95 PC, MS Windows 98 PC, MS Windows NT PC, MS Windows 2000 PC, MS Windows XP	C++	Comercial	Julho/2003
18	SYROCO	Editor Gráfico Simulação Rápida Análise de Desempenho Simples Intercâmbio de Formato de Arquivo Geração de código C++		C++	Comercial	Julho/2004
19	TimeNET	Editor Gráfico Animação com Jogador de Marcas Simulação Rápida Espaço de Estados Invariantes de Lugar Análise Estrutural Análise de Desempenho Simples Análise de Desempenho Avançada Intercâmbio de Formato de Arquivo	Sun Linux		Comercial	Maio/2003
20	Tina	Editor Gráfico	Sun, SunOS		Gratuita	Abril/2005

No.	Nome do Software	Características	Sist. Operacional	Ambiente	Licença	Atualização
		Animação com Jogador de Marcas Espaço de Estados Espaço de Estados Condensado Invariantes de Lugar Invariantes de Transição Classes de Espaço de Estados	PC, Linux PC, MS Windows 98 PC, MS Windows NT PC, MS Windows 2000 PC, MS Windows XP Macintosh, Mac OS X			
21	Visual Object Net ++	Editor Gráfico Animação com Jogador de Marcas Simulação Rápida Análise Estrutural Análise de Desempenho Simples Suporta Hierarquia de Objetos	MS Windows		Gratuita	Maio/2003

Tabela 2.3: Escolha da ferramenta de simulação a ser utilizada neste trabalho

Nome do Software	Pontos (Características)	Pontos (Sist. Operacional)	Pontos (Licença)	Total
JFern	3	5	5	13
Petri Net Toolbox	5	5	3	13
PNtalk	3	5	5	13
Renew	3	5	5	13
Tina	3	5	5	13
Visual Object Net ++	3	5	5	13
JPetriNet	1	5	5	11
PACE	3	5	3	11
PEP	3	3	5	11
Petri .NET Simulator	3	5	3	11
PROTOS	3	5	3	11
StpnPlay	3	5	3	11
TimeNET	5	3	3	11
MISS-RdP	1	5	3	9
Opera	5	1	3	9
PED	1	3	5	9
Romeo	1	3	5	9
Simulaworks	1	5	3	9
SYROCO	3	3	3	9
PetriSim	1	1	5	7
SEA	1	1	5	7

3 Proposta da Dissertação

Neste trabalho de pesquisa, a representação dos estados e das ações dos SFMs são representados graficamente através da modelagem em Rede de Petri Temporizada. A análise de performance do modelo é realizada através da simulação do sistema utilizando as equações de estado, e feita através da quantidade de itens nos *buffers*, disponibilidade dos equipamentos, e quantidade de itens na saída do sistema.

A modelagem também incorpora os conceitos de confiabilidade dos equipamentos/processos, no que se refere à ocorrência de falhas no chão-de-fábrica e, nesta base, este trabalho de pesquisa pretende demonstrar a influência da confiabilidade na performance dos sistemas.

Neste trabalho, para a análise de performance dos sistemas é utilizado uma Rede de Petri p-t-Temporizada (rede temporizada com tempos associados aos lugares e às transições), a qual chamamos de Rede Temporizada Modificada.

A Rede de Petri p-t-temporizada também foi utilizada por Boufaied (2002), no seu trabalho sobre a modelagem de detecção de falhas em sistemas distribuídos através de crônicas, contribuindo para o desenvolvimento de sistemas supervisórios e de monitoração. As crônicas são conjuntos de eventos e restrições temporais entre estes eventos. As crônicas são utilizadas para a detecção de restrições de operação dos processos (ou, formalmente colisões em um determinado instante de tempo) e atrasos nos processos de fabricação. As restrições de tempo são definidas de duas formas: restrições de intervalo, e restrições de uma janela de tempo, onde está incluso um conjunto de restrições de intervalos. E, baseado nestas restrições de tempo, a rede combinada p-t-temporizada é modelada com a restrição do intervalo associada aos lugares e, a restrição de uma janela de tempo associada às transições, implicando que o disparo das transições tem prioridade sobre os disparos dos lugares. Entretanto, Boufaied (2002) não apresenta uma lógica linear para validar este modelo, impossibilitando a simulação do mesmo.

Devido a inexistência de uma ferramenta de simulação para a Rede de Petri p-t-Temporizada, as simulações realizadas para esta rede foram implementadas com o uso do Matlab através do autômato. Neste trabalho a modelagem da Rede de petri p-t-Temporizada foi realizada utilizando-se o modelo temporal de Merlin. A escolha para a realização do modelo temporal de Merlin, em detrimento do modelo de Ramchandani, se deveu aos seguintes fatores:

- O modelo de Merlin tem uma abordagem mais consistente para a representação dos sistemas produtivos, onde os tempos de processo apresentam uma pequena variação a cada ciclo produtivo;
- O modelo de Merlin é equivalente ao modelo de Ramchandani quando consideramos que o par de tempo $\delta = (a,b)$ é tal que *a* = *b*, isto é, quando EFT (*earliest firing time*) é igual a LFT (*latest firing time*);
- Na academia, a tendência de pesquisa evoluiu para a discussão de sistemas temporizados equivalentes aos autômatos temporizados (ALUR & DILL, 1994) já conhecidos, devido a importância de se conceber e implementar um sistema de controle realimentado da

produção. Em Haar (2002) é demonstrado que os autômatos temporizados são equivalentes às Redes de Petri Temporizadas derivada de Merlin.

Para validar este modelo foi utilizada a comparação das classes de estados obtidas com a simulação do autômato através de uma lógica linear e as classes de estados obtidas através da simulação de um modelo equivalente no Tina.

O software de simulação Tina será descrito a seguir.

3.1 Software Tina

Tina (*Time Petri Net Analyser*) é um ambiente de software para edição e análise de Rede de Petri e Rede de Petri Temporizada. Em adição à edição usual e ferramentas de análise, o software Tina oferece várias construções de espaço de estados que preservam as propriedades específicas das redes. Estas propriedades podem ser gerais (propriedades de alcançabilidade, vivacidade, etc.), ou específicas sobre as estruturas lineares dos espaços de estados (propriedades lógicas temporais, teste de equivalência).

Para sistemas temporizados, o Tina utiliza as classes de estados para analisar o seu comportamento. Para sistemas não temporizados, Tina utiliza técnicas de redução baseadas em métodos de ordem parcial para prevenir a explosão combinatorial.

O Tina poderia ser tipicamente utilizado como um verificador de modelo, provendo espaços de estados reduzidos, sobre os quais as propriedades desejadas podem ser checadas mais eficientemente do que no espaço de estado original. Dentro de sistemas concorrentes, as técnicas usadas às vezes resultam em uma grande redução do tamanho nos espaços de estados. O domínio de aplicação do Tina é amplo (BERTHOMIEU ET AL., 2004).

As funcionalidades do Tina são implementadas modularmente. Estes módulos podem ser usados independentemente ou em combinação. Os módulos incluem:

- Um editor gráfico para Rede de Petri, Rede de Petri Temporizada, ou autômatos;

- Uma ferramenta para a construção de abstrações de espaço de estados;
- Uma ferramenta de análise estrutural (ainda em desenvolvimento);

O editor produz arquivos que podem ser lidos através da construção do espaço de estados e das ferramentas de análise estrutural.

As ferramentas de construção e de análise funcionam como “filtro”. Também admitem que as descrições de entrada estejam no formato gráfico ou formato texto, e ainda podem produzir os resultados em uma variedade de formatos.

A Figura 3.1 abaixo mostra uma tela típica do Tina, com uma Rede de Petri editada, o resultado textual de um comportamento estrutural, e o resultado textual das classes de estados obtidas.

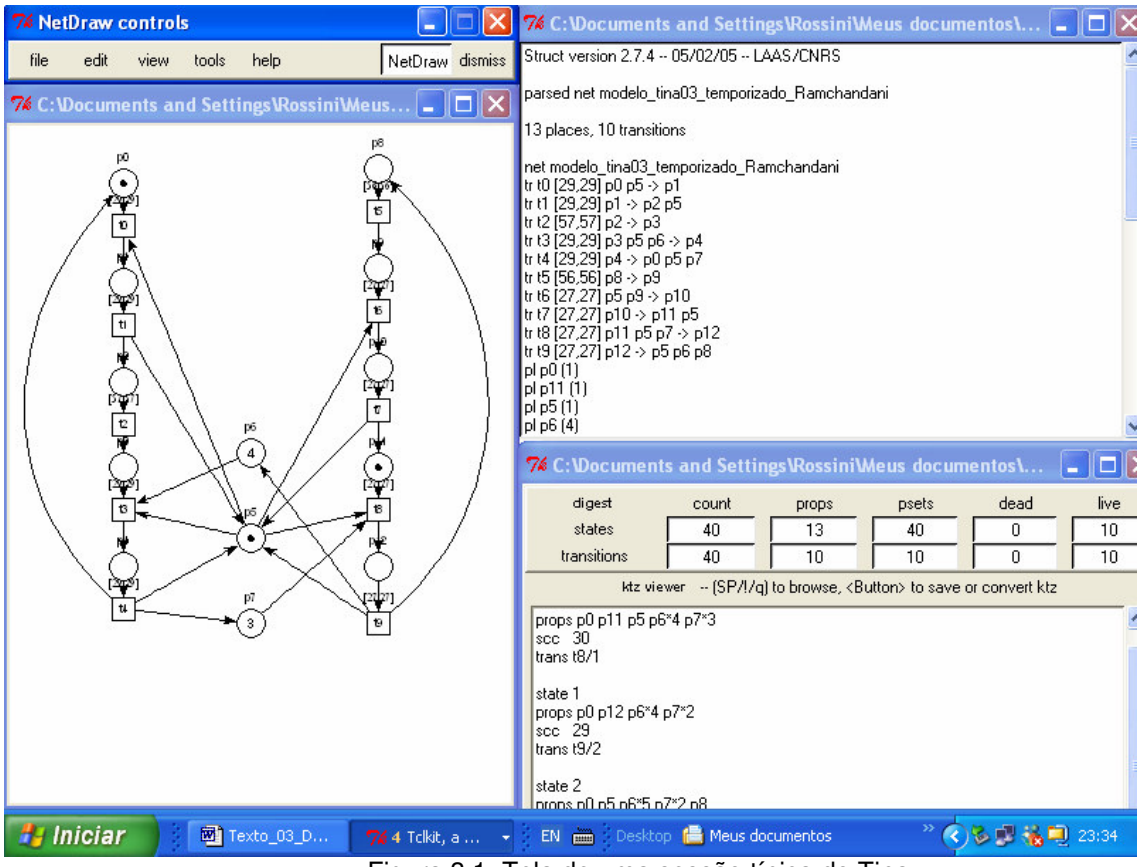

Figura 3.1: Tela de uma sessão típica do Tina.

3.2 Rede Temporizada – Exemplo de Aplicação

Na Figura 3.2 é mostrado um exemplo de aplicação com a ferramenta de simulação do Tina. Este exemplo consiste num esquema produtor – consumidor,

através de duas máquinas, buffer intermediário com esquema de exclusão mútua feito através de um robô de manipulação.

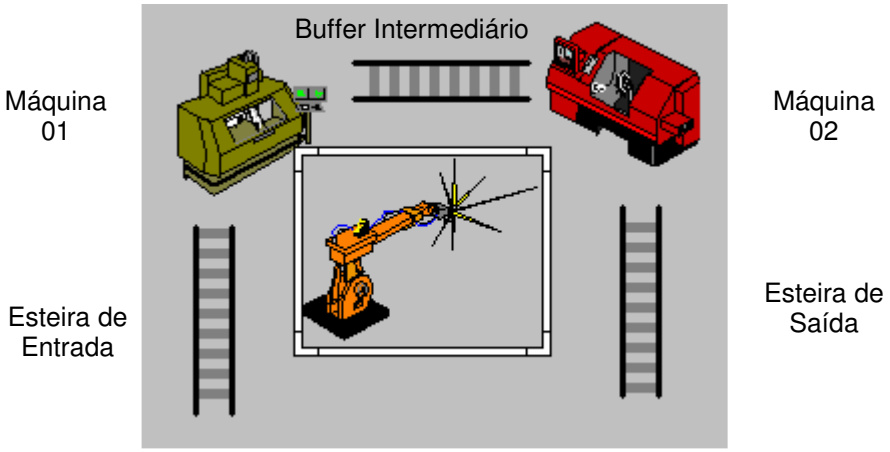

Figura 3.2: Ilustração do esquema produtor – consumidor.

As peças brutas (matéria-prima) chegam a partir da esteira de entrada. A chegada da peça é detectada por um sensor.

Quando uma peça bruta está presente, a máquina 01 não está carregada e o robô livre, então se dá início ao processo de carregamento. A máquina 01 executa a operação e espera o descarregamento pelo robô para o depósito da peça processada no buffer. O depósito é feito quando existe um espaço no *buffer* e o robô está novamente livre. De forma análoga, é realizado o processo em relação a máquina 02 (DICESARE & HARHALAKIS, 1993).

Neste exemplo, os lugares e transições são assim definidos: p0 significa máquina 01 pronta e peça bruta na esteira; p1, máquina 01 carregada; p2, operação 01 iniciada; p3, máquina 01 esperando descarregamento da peça; p4, descarregamento realizada; p5, robô livre; p6, buffer de peças; p7, *buffer* de peças intermediárias; p8, operação 02 iniciada; p9, máquina 02 esperando descarregamento; p10, descarregamento iniciado; p11, máquina 02 esperando peça da operação 01; p12, máquina 02 carregada; t0 significa autorização de carregamento; t1, autorização de início da operação da máquina 01; t2, final da operação 01; t3, autoriza início de descarregamento; t4, final da operação 01; t5, final de operação da máquina 02; t6, autorização início de descarregamento da máquina 02; t7, fim de depósito; t8, início de carregamento da máquina 02; e t9, início de operação 02.

A rede de Petri para este esquema é apresentada na Figura 3.3 abaixo:

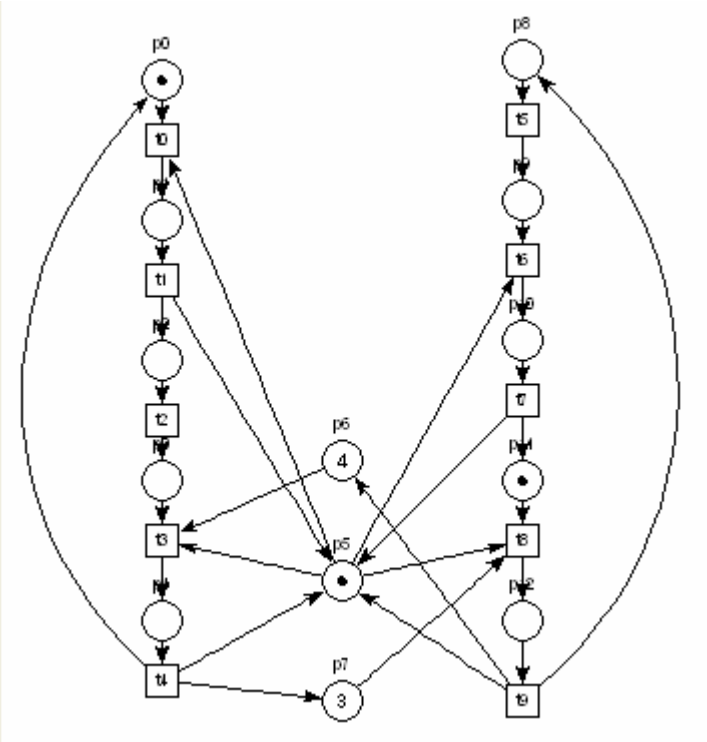

Figura 3.3: Rede de Petri do esquema produtor – consumidor.

3.3 Proposta de Tratamento da Confiabilidade nas Redes de Petri Temporizadas

Para tornar mais realística a abordagem feita neste trabalho, passou-se a considerar as interrupções nos processos por falha dos equipamentos e, assim, considerar a dinâmica real do chão-de-fábrica, no que se refere às perdas no sistema produtivo. Neste cenário, a performance do sistema é medido essencialmente pela freqüência de duração com que cada equipamento ou processo encontra-se disponível para a operação. Isto sugere a utilização de um modelo estocástico simples para cada equipamento ou processo, limitado a dois estados, conforme representado na Figura 3.4, a seguir:

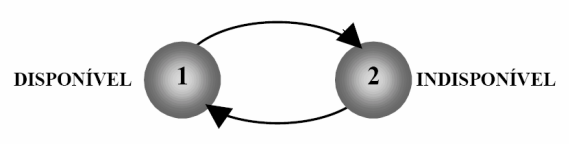

Figura 3.4: Modelo de Disponibilidade.

Esta abordagem também é realizada por DiCesare & Harhalakis (1993), onde as características de confiabilidade são representadas através dos parâmetros MTBF (*Medium Time Between Failures* – tempo médio entre falhas) e MTTR (*Medium Time To Repair* – tempo médio para o reparo), em uma configuração mostrada na Figura 3.5 abaixo:

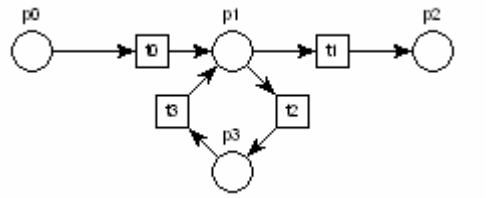

Figura 3.5: Trecho de uma rede de Petri, para a representação do tratamento da confiabilidade dos equipamentos.

Na Figura 3.5 acima se pode observar que para cada equipamento ou processo em que seja considerada a característica de confiabilidade, serão adicionados 02 transições e 01 lugar à rede original.

O fato acima aumenta o tamanho da rede, podendo tornar-se crítico para sistemas mais complexos, uma vez que dificulta o entendimento da representação gráfica, uma vez que descaracteriza a representação mais simples e usual do processo modelado, onde não há adição dos lugares e transições conforme exposto acima.

Por outro lado, esta modelagem implica em dizer que a marcação (ficha ou *token*) não estará disponível após se atingir o tempo característico de MTBF do equipamento ou processo, somente retornando depois de percorrido o período de tempo correspondente ao MTTR. Assim, pode-se dizer que os parâmetros MTBF e MTTR são característicos dos lugares (relacionados aos equipamentos/processos), e a sua dinâmica irá representar uma alteração no vetor de marcação, na matriz de incidência e, conseqüentemente, no vetor de habilitação.

Esta nova abordagem, a Rede de Petri p-t-Temporizada, não necessita a adição de lugares e transições, e é apresentada a seguir.

Definição 1: Uma rede t-temporizada modificada é um par $\langle R, I_p \rangle$, onde:

- *R* é uma rede t-temporizada
- I_p é uma aplicação definida como
  - $I_p : P \rightarrow \left(Q^+ \cup \{0\}\right) \times \left(Q^+ \cup +\infty\right)$ (19)
  - $p_i \rightarrow I_{pi} = \left[MTBF_i, MTTR_i\right]$ (20)

A aplicação l_p associa uma função, ou *n* funções, com duração de tempo racional e positivo associado a cada lugar no qual se deseja icorporar as carcterísticas de confiabilidade do equipamento/processo que representa, e que pode ser representada pelos modelos de Ramchandani ou Merlin. Ou seja, esta aplicação está associada à dinâmica das fichas (*tokens*) com o tempo.

As funções definidas por $MTBF_i$ e $MTTR_i$ representam, respectivamente, o tempo no qual todas as fichas (*tokens*) presentes no lugar relacionado ao equipamento p_i serão removidas virtualmente e, depois retornarão ao mesmo lugar relacionado ao equipamento para a continuidade do processo segundo a dinâmica da Rede de Petri t-Temporizada.

De forma genérica, temos que cada uma destas funções usa dois números reais, *c* e *d*, com $c \leq d$, que são associados a cada lugar relacionado ao equipamento p_i através da respectiva função.

i. *c* ($0 \leq c$) é o mínimo tempo que deve decorrer, a partir do tempo no qual o lugar relacionado ao equipamento p_i está habilitado, até poder ser disparado, de acordo com a respectiva função;
ii. *d* ($0 \leq d \leq \infty$) denota o tempo máximo de duração na qual o lugar relacionado ao equipamento p_i pode estar habilitado sem ter sido disparado.

De forma similar os tempos *c* e *d*, para uma dada função associada ao lugar relacionado ao equipamento p_i, são relativos ao momento no qual o lugar p_i está habilitado. Assumindo que um lugar relacionado ao equipamento p_i que contém a aplicação l_p foi habilitado no tempo φ, então p_i, se continuamente habilitado, não pode disparar antes do tempo $\varphi + c$ e deve disparar antes ou no tempo $\varphi + d$, a menos que seja desabilitado antes do disparo pelo disparo de alguma outra transição concorrente.

Cada lugar disparado “virtualmente” no intervalo de tempo associado, implica dizer, que cada ficha (*token*) deve ser processada de acordo com as regras da função associada no intervalo de tempo associado. Desta forma, neste trabalho, para as funções definidas, temos que:

iii. A função $MTBF_i$ é habilitada com a presença de uma ou mais fichas (tokens) no lugar relacionado ao equipamento e, o relógio somente é resetado após a ocorrência do disparo “virtual”. Caso uma transição seja disparada anteriormente e se todas as fichas (*tokens*) presentes no lugar

sejam consumidas, o relógio deverá ter sua contagem somente interrompida, até que o disparo seja efetivado e o relógio local resetado.

iv. A função $MTBF_i$ quando habilitada e disparada no lugar relacionado ao equipamento p_i, no intervalo de tempo associado, consome todas as fichas (tokens) retirando-as virtualmente do lugar p_i .

v. A função $MTTR_i$ quando habilitada e disparada no lugar relacionado ao equipamento p_i, no intervalo de tempo associado, retorna todas as fichas (tokens) ao p_i , antes do disparo da função $MTBF_i$.

vi. A função $MTTR_i$, somente pode ser habilitada no lugar relacionado ao equipamento p_i após a função $MTBF_i$ já ter sido habilitada e disparada neste mesmo lugar. Por outro lado, a função $MTBF_i$ somente pode ser habilitada e disparada no lugar relacionado ao equipamento p_i uma única vez, exceto se a função $MTTR_i$ tiver sido disparada anteriormente neste lugar.

vii. A função $MTTR_i$ é habilitada automaticamente após a função $MTBF_i$ c ter sido disparada e a contagem do relógio é iniciada, somente sendo resetado após a ocorrência do disparo “virtual”, quando todas as fichas “virtualmente” removidas retornam “fisicamente” ao lugar original.

Definição 2: Formalmente uma Rede de Petri Temporizada Modificada é uma n-upla $PN = (P, T, F, W, M_0, \delta, \varepsilon_{MTBF}, \varepsilon_{MTTR})$, onde:

- $P = \{p_1, p_2, \cdots, p_m\}$ é um conjunto finito de lugares;
- $T = \{t_1, t_2, \cdots, t_n\}$ é um conjunto finito de transições;
- $F \subseteq (P \times T) \cup (T \times P)$ é um conjunto de arcos orientados;
- $W : F \to \{1,2,3,\cdots\}$ é a função peso;
- $M_0 : P \to \{0,1,2,3,\cdots\}$ é a marcação inicial;
- $\delta : T \to Q^+ \times (Q^+ \cup \infty)$, onde Q^+ é conjunto dos números racionais positivos;
- $\varepsilon_{MTBF} : P \to Q^+ \times (Q^+ \cup \infty)$, onde Q^+ é conjunto dos números racionais positivos, associada à função $MTBF_i$;
- $\varepsilon_{MTTR} : P \to Q^+ \times (Q^+ \cup \infty)$, onde Q^+ é conjunto dos números racionais positivos, associada a função $MTTR_i$;
- $P \cap T = \phi$ e $P \cup T \neq \phi$

Na Rede de Petri p-t-Temporizada existe uma concorrência entre os disparos das transições e os disparos nos lugares. Portanto, é caracterizado um conflito estrutural "implícito", no qual se faz necessário uma regra de prioridade para o estabelecimento da dinâmica deste modelo. Basicamente, o que temos é uma estrutura de rede reconfigurável quando da ocorrência dos disparos associados aos lugares pelas funções ε_{MTBF} e ε_{MTTR}, devido a alteração ocorrida no vetor de marcação e na matriz de incidência, simultâneamente. Esta reconfiguração de rede acontece devido a eliminação virtual do lugar quando habilitado o disparo da função $MTBF_i$. Considerando o modelo da Figura 3.5, a Figura 3.6 representa a seqüência de eventos que se daria em uma Rede de Petri, no processo equivalente ao ocorrido na Rede de Petri p-t-temporizada, contendo como exemplo o disparo da função $MTBF_i$ no lugar p_1, a reconfiguração da rede (conforme discutida acima) e a sua restauração após o disparo da função $MTTR_i$ no lugar p_3 adicionado para a representação do sistema em falha.

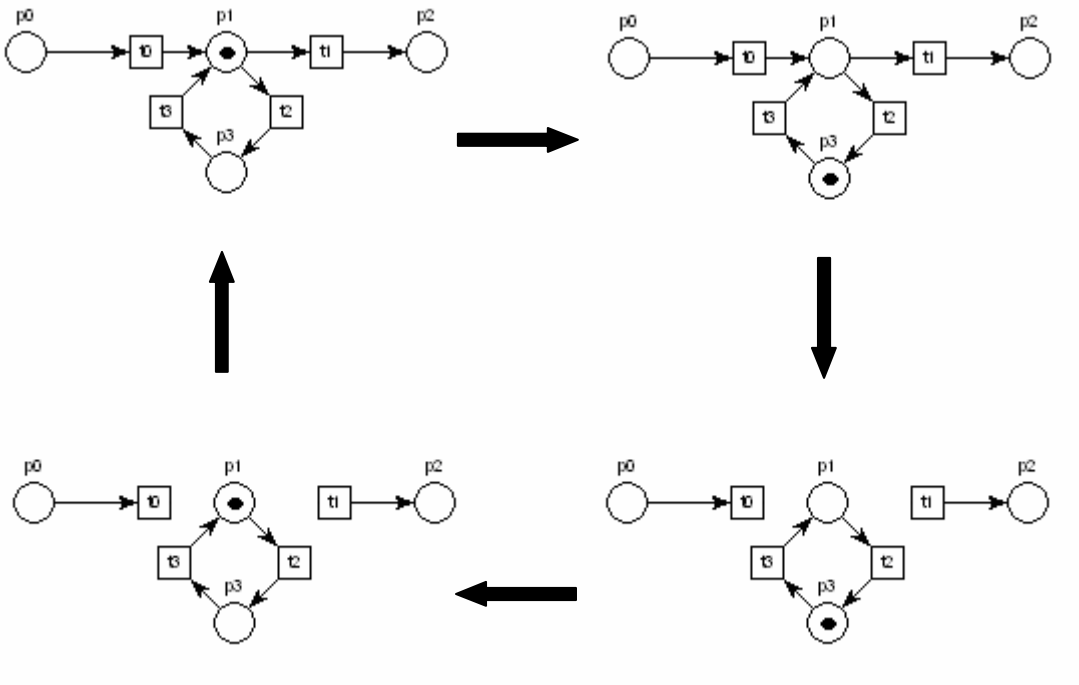

Figura 3.6: Seqüência de eventos em uma Rede de Petri, para a representação do disparo das funções $MTBF_i$ e $MTTR_i$.

A Figura 3.7 mostra a seqüência de eventos na Rede de Petri p-t-Temporizada tendo o exemplo do disparo da função $MTBF_i$ no lugar p_1, a

reconfiguração da rede, e a sua restauração após o disparo da função $MTTR_i$ no mesmo lugar (p_1) restaurando a rede inicial. A reconfiguração da rede é necessária para se evitar a ocorrência de disparos do lugar p_0 para o lugar p_1, uma vez que o lugar p_1 se encontra em falha e o processo está interrompido. A retirada das fichas do lugar p_1 é dita "virtual" devido as fichas somente serem removidas do lugar, não sendo inserida em nenhum outro lugar da rede. A Figura 3.6 e Figura 3.7 são equivalentes.

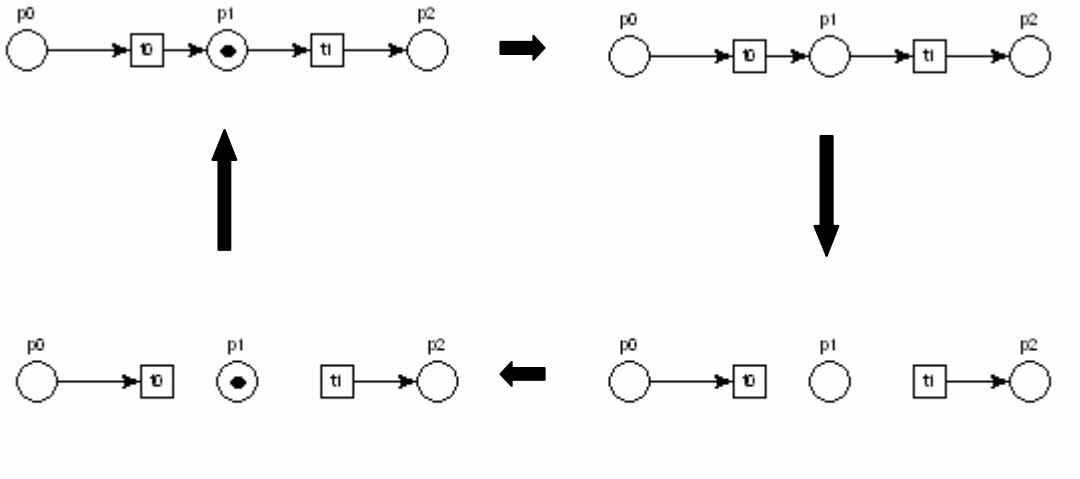

Figura 3.7: Seqüência de eventos em uma Rede de Petri p-t-Temporizada, para a representação do disparo das funções $MTBF_i$ e $MTTR_i$.

Antes da aplicação $MTBF_i$ estar habilitada, as fichas presentes no lugar relacionado ao equipamento não sofrem nenhuma alteração e, portanto, o vetor marcação continua o mesmo do modelo da rede inicial. Após o período de tempo em que a aplicação $MTBF_i$ é disparada, o que representa a indisponibilidade ou falha do equipamento ou processo, todas as fichas são removidas e, portanto, o vetor de marcação também deve ser modificado para caracterizar esta ausência. A matriz de incidência também é modificada através da remoção de todas as transições associadas ao lugar relacionado ao equipamento, estando em conformidade com o ponto de vista do processo, onde a aplicação $MTBF_i$ também representa a interrupção do fluxo.

A modificação da matriz de incidência, isto é, a eliminação de todas as transições, é feita através de um algoritmo polinomial, zerando-se todos os elementos da linha correspondente ao lugar no qual foi disparado a função MTBF.

A aplicação $MTTR_i$, representa a restauração (ou reparo) do estado de falha do equipamento ou processo. Neste caso a aplicação $MTTR_i$, somente pode ser habilitado após a aplicação $MTBF_i$ ter concluído o disparo e, não ter fichas no lugar relacionado ao equipamento correspondente. Após o período de tempo em que a aplicação $MTTR_i$ é disparada, o que representa a restauração da disponibilidade ou falha do equipamento ou processo, todas as fichas retornam ao lugar e, portanto, o vetor de marcação também deve ser modificado para caracterizar esta recomposição do estado anterior à falha. Do ponto de vista do processo, a aplicação $MTTR_i$ também representa a restauração da continuidade do fluxo do processo. Esta restauração do fluxo do processo, do ponto de vista formal da Rede de Petri, representa o retorno do lugar relacionado ao equipamento e das transições a eles associadas, isto é, a restauração da rede original obtida pela desconfiguração da rede modificada, ou seja, a restauração da modificação feita na matriz de incidência da rede.

A dinâmica da evolução da Rede Temporizada Modificada depende da marcação da rede e da situação temporal das fichas em relação aos lugares e transições associadas.

A regra de disparo da Rede de Petri Temporizada Modificada é uma conjunção das regras de disparo das Redes de Petri T-Temporizadas e P-Temporizadas, adicionando-se a regra de decisão (ou de prioridade) para a eliminação do conflito estrutural existente.

A regra de disparo do lugar é formalmente expressa pelas seguintes condições:

1. Satisfaz a todas as condições de disparo de transição de uma Rede de Petri;
2. Somente pode ser disparada se existe ao menos uma ficha no lugar correspondente;
3. Expressa o fato que o lugar ou transição habilitada não pode disparar antes de EFT e deve disparar antes ou em LFT a menos que outros disparos ocorram modificando os estados atuais.
4. Assim, o lugar p_i é disparável a partir de um estado S no tempo $\tau + \varphi$ se as seguintes condições são satisfeitas:
    - p_i foi habilitada pela marcação M no tempo τ;

- O tempo relativo de disparo φ, relativo ao tempo de disparo absoluto τ, não é menor que EFT do lugar p_i e não é maior que os LFTs de todas as transições habilitadas pela marcação M, isto é, EFT de $p_i \leq \theta \leq$ min{LFT de t_k }, onde k é a cardinalidade do conjunto de lugares habilitadas por M.
- Não existe uma segunda aplicação em que o lugar também está habilitado para o disparo (restrição de aplicação);
- O disparo de um lugar não pode ser interrompido e, também não se deve permitir que o lugar inicie um segundo disparo antes do primeiro estar finalizado;
- O tempo de atraso φ não é um tempo global, e pode ser visto como um relógio local e virtual para cada lugar, que deve estar na mesma escala de tempo para todas os lugares da rede;
- Em caso de conflito no disparo do lugar ou da transição, diferentemente do proposto por Boufaied (2002), a prioridade deve ser dada ao lugar;

5. Assim, a transição t_i é disparável a partir de um estado S no tempo $\tau + \theta$ se as seguintes condições são satisfeitas:
   - t_i foi habilitada pela marcação M no tempo τ;
   - O tempo relativo de disparo θ, relativo ao tempo de disparo absoluto τ, não é menor que EFT da transição t_i e não é maior que os LFTs de todas as transições habilitadas pela marcação M, isto é, EFT de $t_i \leq \theta \leq$ min{LFT de tk }, onde k é a cardinalidade do conjunto de transições habilitadas por M.
   - O tempo de atraso θ não é um tempo global, e pode ser visto como um relógio local e virtual para cada transição, e que deve estar na mesma escala de tempo para todas as transições na Rede de Petri Temporizada.
   - O disparo de uma transição não pode ser interrompido e, também não se deve permitir uma transição iniciar um segundo disparo antes do primeiro estar finalizado.

- Não pode ser disparada em conflito com o disparo de um lugar (de pré-condição ou pós-condição) sem que ocorra antes a modificação da matriz de incidência.

Baseado na equação de estado definida em (16), temos a seguinte equação de estado para a Rede Temporizada Modificada:

$$M_{k+1} = M_{km} + A_m{}^t v_{kt} \tag{21}$$

Onde:

- M_{km}, é o vetor coluna de marcação modificado, dado por:
  - $$M_{km} = \begin{cases} M_k, se\ \varphi_i \le MTBF_i\ ou\ \varphi_i \ge MTTR_i \\ 0, se\ MTBF_i \le \varphi_i \le MTTR_i \end{cases} \tag{22}$$
- A_m^t, é a matriz de incidência modificada, dada por:
  - $$A_m^t = \begin{cases} A^t, se\ \varphi_i \le MTBF_i\ ou\ \varphi_i \ge MTTR_i \\ A^t, com\ A^t(i,k) = 0, se\ MTBF_i \le \varphi_i \le MTTR_i \end{cases} \tag{23}$$
  - Onde *k*, representa o índice correspondente a todas as transições da rede.

Devido às modificações que ocorrem nos vetores de marcação e, na matriz de incidência, que provocam uma nova marcação da rede e a reconfiguração desta, na Rede de Petri Temporizada Modificada, surge uma peculiaridade que se traduz na seguinte regra: o estado seguinte, M_{k+1} não pode ser alcançável caso exista uma modificação no vetor de marcação M_{km} e/ou na matriz de incidência A_m^t, provocados pelo disparo da aplicação I_p associada a algum lugar da rede.

3.4 Simulação da Rede Temporizada Modificada

Atualmente, não é possível realizar a modificação da ferramenta de simulação Tina para a Rede Temporizada Modificada apresentadas acima. Assim, para validar a proposta apresentada utilizou-se o seguinte método:

1. Escolher um exemplo para validar o modelo. Neste caso, foi escolhido o exemplo da Figura 3.2;
2. Realizar a modelagem do exemplo escolhido no Tina;

3. Implementação do algoritmo utilizando o ambiente de programação do Matlab;
4. Executar a simulação no Tina;
5. Executar a simulação no Matlab;
6. Comparar os resultados do Tina com o Matlab, através das classes de estados e matriz de alcançabilidade, através do modelo LSCG (*Linear state classes with multi-enabledness construction*);
7. Validar através da compatibilidade das saídas, isto é, o autômato simulado no Matlab é equivalente à Rede modelada no Tina;

Esta seqüência é mostrada no fluxograma da Figura 3.8 a seguir. Neste processo de validação o Tina é utilizado como verificador de modelo.

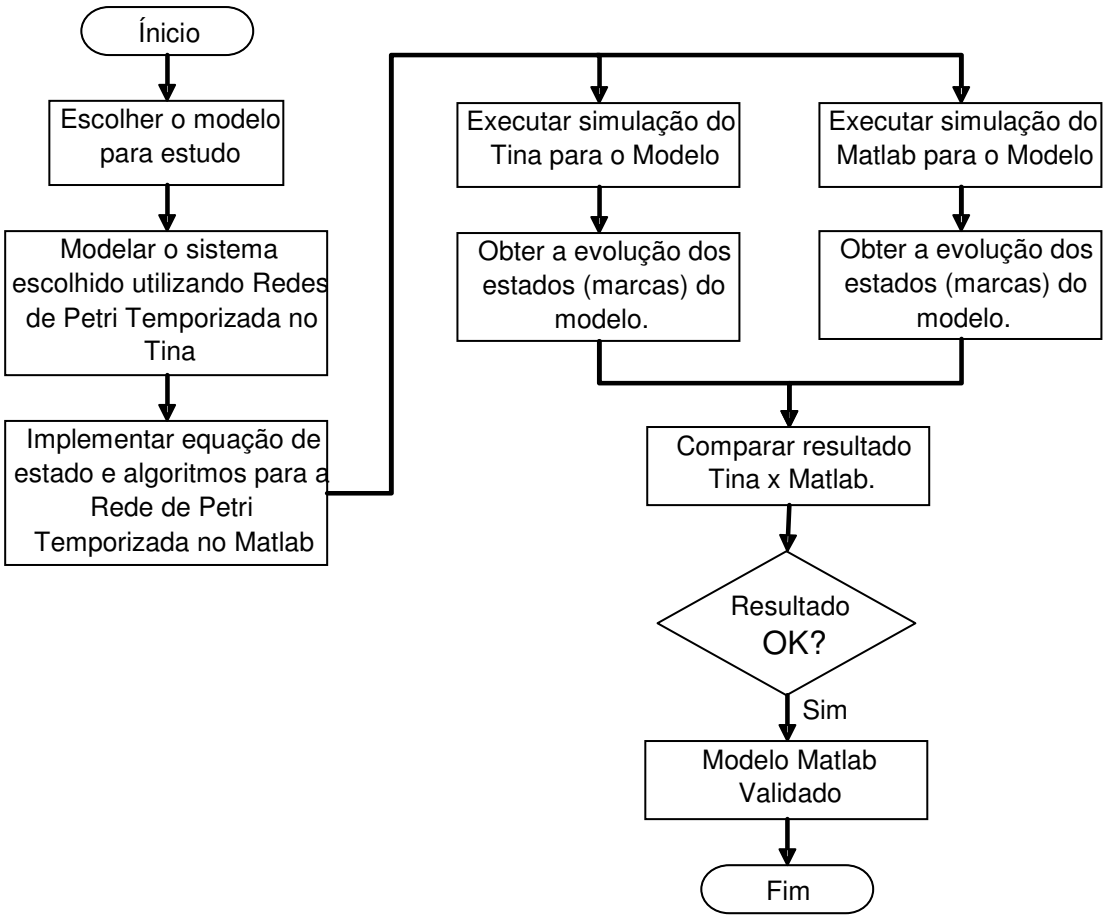

Figura 3.8: Fluxograma de Validação do Algoritmo no Matlab.

Para a implementação da dinâmica de evolução das marcas (estados) dos sistemas, é necessário um algoritmo para calcular o vetor de habilitação. O algoritmo utilizado neste trabalho foi proposto em Del Foyo (2001) e é apresentado a seguir.

3.4.1 Algoritmo para o Cálculo do Vetor de Habilitação

Para o cálculo do vetor de habilitação, foi utilizado o algoritmo apresentado e utilizado por Del Foyo (2001) para a equação de estado da rede GHENeSys, modificado apropriadamente para o caso em estudo. A seguir é mostrado o algoritmo.

O vetor de habilitação é determinado após os seguintes passos:

i. Primeiramente, multiplica-se o vetor de macacão (M_k) por uma matriz linha composta de 1s, de ordem 1xn, onde n é o número de transições da rede.
ii. O resultado desta multiplicação é uma matriz de ordem (mxn), ou seja, da mesma ordem que a matriz de incidência A, e a qual denominaremos M_a.
iii. Em seguida, soma-se as matrizes M_a e A, resultando na matriz que é denominada como M_i.

A matriz M_i é obtida pela seguinte equação:

$$M_i = M_k I + A^t \quad (24)$$

Cada coluna de M_i denomina-se como v_i e, representa o disparo resultante de cada atividade a_i. Portanto para determinar se esta atividade está habilitada deve-se comparar esta possível marcação resultante com uma marcação válida de acordo com a capacidade do sistema.

Uma atividade *a* está habilitada se, e somente se, o seu vetor coluna correspondente v_i satisfaz à desigualdade $v_0 \le v_i \le K^t$, onde:

iv. v_0 é um vetor da mesma ordem de v_i com todos os seus elementos iguais a zero;
v. K é o vetor das capacidades;

Para a verificação da desigualdade a seguinte definição foi utilizada:

- Um vetor X é maior ou igual a um vetor Y (X≥Y) se e somente se:
  - X e Y são da mesma ordem;
  - O elemento *i* de X é maior ou igual ao elemento *i* de Y: $x_i \ge y_i$, onde $1 \le i \le z$, onde *z* é o número de elementos de X e Y.

PROPOSIÇÃO 01: Uma atividade é disparável se e somente se:

- O seu vetor coluna correspondente v_i satisfaz a desigualdade $v_0 \le v_i \le K^T$, onde v_0 é um vetor da mesma ordem de v_i com todos os seus elementos iguais a 0 (zero), e K é o vetor das capacidades.
- o tempo relativo de disparo φ, relativo ao tempo de disparo absoluto τ, não é menor que EFT do lugar a_i e não é maior que os LFTs de todas as transições habilitadas pela marcação M, isto é, EFT de $a_i \le \theta \le$ min{LFT de t_k }, onde k é a cardinalidade do conjunto de atividades habilitadas pela marcação M.

Porém, este vetor não indica que as atividades habilitadas possam acontecer simultaneamente, pois não poderão caso ocorra uma situação de conflito ou contato na rede. Nestas situações é preciso saber quais são as atividades envolvidas e, estabelecer uma regra de decisão, que pode ser determinística ou aleatória, para resolver a situação de contato ou conflito. Neste trabalho foi utilizada uma regra de decisão determinística.

Para a detecção de conflitos e contatos, foi utilizado o algoritmo apresentado e utilizado por Del Foyo (2001), sendo necessário quando se considera um sistema que dispara o maior número possível de atividades, uma vez que estejam habilitadas. A seguir é mostrado o algoritmo.

Para detectar situações de conflito ou contato, primeiramente devemos calcular o vetor de habilitação utilizando o algoritmo apresentado anteriormente. Depois, o segundo passo, é calcular a matriz de conflito, denominada M_c, que é uma matriz coluna de ordem mx1. Esta matriz é obtida pela seguinte equação:

$$M_c = M_k + A^t v_k \quad (25)$$

A seguir, é obtido o vetor de conflito, denominado de v_c, determinado pela seguinte regra:

$$[v_c]_{j1} = \begin{cases} -1 \; se \; [M_c]_{j1} < 0 \\ 1 \; se \; [M_c]_{j1} > K(j) \\ 0 \; demais \; casos \end{cases} \quad (26)$$

E, finalmente, uma vez calculado o vetor v_c é possível determinar quais dos vetores habilitados estão em conflito, determinando-se a matriz C_c segundo a seguinte equação:

$$C_c = A^t + (v_k v_c) \quad (27)$$

Os conflitos são determinados usando a seguinte definição:

- Dada a matriz de conflito C_c:
  - Para um lugar p_i, as atividades a_j, tais que, $[C_c]_{ij} = -2$ estão em conflito devido ao compartilhamento do lugar;
  - Para um lugar p_i, as atividades a_j, tais que, $[C_c]_{ij} = 2$ estão em conflito devido ao compartilhamento dos pós-elementos;
  - Nos demais casos, $[C_c]_{ij} = [v_k]_{1j}[v_c]_{1i}$.

3.4.2 Simulação: Exemplo Sistema Produtor - Consumidor

Para o sistema produtor – consumidor apresentado na Figura 3.2, e levando-se em consideração o modelo em Rede de Petri Temporizada da Figura 3.3, a matriz de incidência, vetor de capacidade e o vetor de marcação inicial são apresentadas, respectivamente, pelas equações (28), (29) e (30), abaixo:

$$A = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & -1 & 1 & 0 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \quad (28)$$

$$K = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 7 \\ 7 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{29}$$

$$M_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 4 \\ 3 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{30}$$

Neste mesmo sistema produtor- consumidor, a exemplo do funcionamento do que acontece com o vetor de marcação e a matriz de incidência após a aplicação das funções $MTBF_i$ e $MTTR_i$, considerando que o vetor de marcação do k-ésimo estado é dado pela equação (31), a função $MTBF_i$ estará habilitada e quando da ocorrência do disparo (após o tempo definido para o MTBF e considerando que a transição t_2 não dispare antes disto), o novo vetor de marcação e a matriz de incidência no estado seguinte são dados pelas equações (32) e (33).

$$M_k = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 4 \\ 3 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{31}$$

$$M_{k+1} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 4 \\ 3 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{32}$$

$$A_{k+1} = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & -1 & 1 & 0 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \quad (33)$$

O programa desenvolvido para realizar a simulação no Matlab, considerando a abordagem de Rede de Petri Temporizada Modificada, para este caso é apresentado no Apêndice A.

Para a comparação do algoritmo implementado no Matlab, a Figura 3.9, abaixo apresenta a modelagem no Tina, com os lugares e transições adequadamente acrescentados para o tratamento da confiabilidade relacionada a máquina 01, máquina 02 e robô de manipulação, segundo a abordagem de DiCesare & Harhalakis (1993).

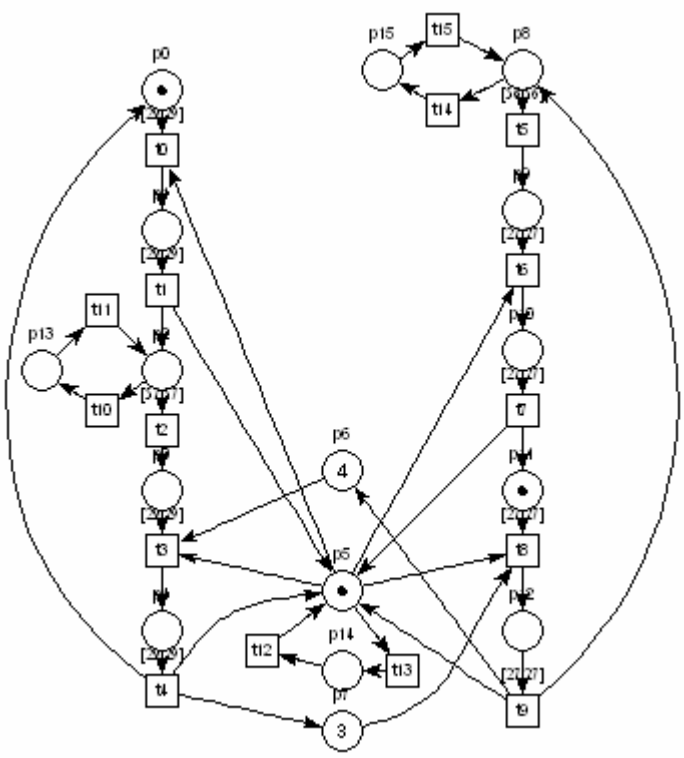

Figura 3.9: Rede de Petri do esquema produtor- consumidor com as características de confiabilidade.

3.4.3 Resultados da Simulação: Exemplo Sistema Produtor - Consumidor

Comparando-se as saídas do programa desenvolvido no Matlab com as saídas do Tina, observa-se que as saídas entre o autômato correspondente a Rede de Petri p-t-Temporizada e a sua modelagem no Tina são compatíveis para o sistema estudado, com exceção da evolução das marcas no período inicial que é devido a diferença da regra de decisão utilizada. Assim, obtemos uma rede que trata das características de confiabilidade preservando a simplicidade da representação gráfica do modelo, uma vez que não é necessário adicionar novos elementos de rede.

Haja visto a validação da modelagem da Rede de Petri p-t-Temporizada, novos algoritmos para a análise de desempenho do sistema foram implementados, habilitando o programa a mensurar as taxas de utilização do equipamento, os tempos de espera, os tempos de bloqueio e os tempos de falha.

Por exemplo, no sistema da Figura 3.2, sem falhas nos equipamentos, e considerando os tempos de transição definidos no programa, tem-se que:

1. O robô de manipulação possui uma taxa de utilização de 48,72%;
2. A máquina 01 possui uma taxa de utilização de 16,11%;
3. A máquina 02 possui uma taxa de utilização de 18,67%;

Por outro lado, para o mesmo sistema considerando o MTBF equivalente a 1/3 do tempo de simulação para cada equipamento (máquina 01, robô de manipulação e máquina 02) e MTTR equivalente a 5% do tempo total de simulação, tem-se os seguintes resultados:

4. O robô de manipulação possui uma taxa de utilização de 50,72%;
5. A máquina 01 possui uma taxa de utilização de 16,11%;
6. A máquina 02 possui uma taxa de utilização de 15,71%;

Onde nota-se uma redução da eficiência na saída do processo, isto é, na máquina 02.

Desta forma, a análise de performance do sistema é feita através dos indicadores acima citados, ou outros indicadores semelhantes que se julguem necessário, sendo possível avaliar a estratégia de programação dos SFMs, pela maximização da saída ou a robustez do sistema em cumprir o plano elaborado.

Com a modelagem utilizando a Rede de Petri p-t-Temporizada é possível obter dados quantitavos sobre a previsão do comportamento do sistema, oferecendo informações consistentes e objetivas para a tomada de decisões para diversos cenários que se necessite realizar a simulação.

4 Estudo de Caso na Indústria Automobilística

4.1 Apresentação de um Problema Real – Indústria Automobilística

Para melhor ilustrar a importância da utilização da Rede de Petri p-t-Temporizada Modificada para a análise de performance através da simulação, neste trabalho foi utilizado como exemplo uma a aplicação prática em um sistema produtivo do ramo automobilístico.

A motivação para a escolha desta aplicação se deu a partir dos dados do histórico de perdas de produtividade de uma indústria automobilística situada em Camaçari-BA no período de maio a dezembro de 2004. Nesta indústria são produzidos três modelos de carro com todas as suas variantes de montagem, de

acordo com a solicitação do cliente (escolha dos itens opcionais, tipo de motor, etc.). A capacidade da fábrica corresponde a 56 carros/hora.

Os sistemas de manufatura são sistemas hierárquicos que, em geral, compreendem a seguinte estrutura hierárquica (MARTINEZ, 2002):

- Nível de fábrica: composto por linhas de produção. Sendo uma linha para cada produto, totalmente ou parcialmente integrada;
- Nível de Linha de Manufatura: composto por células de produção, onde é realizada uma etapa do processo produtivo;
- Nível de Célula de Manufatura: composto por diversos equipamentos;
- Nível de Equipamento: composto por diversos dispositivos e recursos;
- Nível de Operação: composto pelas funções de cada um dos dispositivos.

Seguindo esta estrutura hierárquica, no nível de fábrica, o fluxo do processo produtivo de uma indústria automobilística pode ser representado pela Figura 4.1, onde é mostrado o fluxo dos itens desde o ínicio, na Estamparia, onde são cortados os *blanks*, passando pela Fábrica de carrocerias, onde é fabricado a carroceria do veículo (*body-in-white*) que são pintadas na Pintura e, finalmente chega a Linha de Montagem onde recebe o motor, painel de instrumentso,pneus, e todos os demais acessórios para se ter o carro completo, como chega às concessionárias.

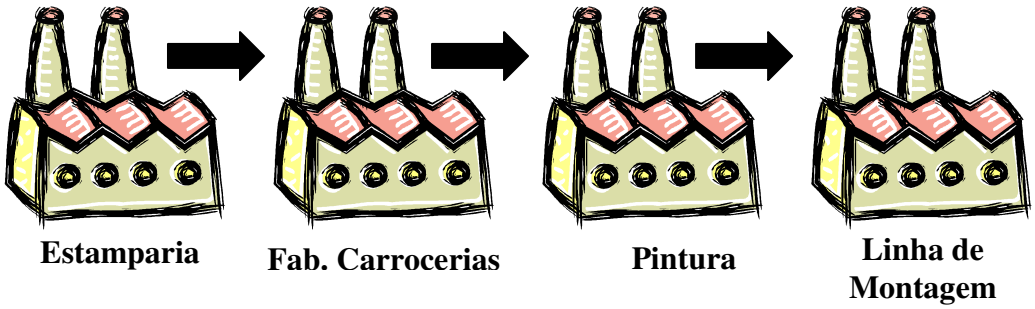

Figura 4.1: Fluxo de processos produtivos, no nível de fábrica (resumido), de uma indústria automobilística.

Seguindo a estrutura hierárquica do sistema produtivo, e tomando como foco a fábrica de carrocerias, temos no nível de linha de manufatura (produção) a seguinte configuração apresentada na Figura 4.2. A fabricação da carroceria (*body-in-white*) é a soldagem e montagem geométrica dos diversos elementos constituintes das partes inferiores, partes superiores e partes móveis. As partes inferiores são

formadas por todos os itens que constituem o compartimento do motor, assoalho inferior e assoalho superior que formam o conjunto assoalho completo. As partes superiores são formadas por todos os itens que constituem a fabricação das laterais das carrocerias. As partes móveis são formadas por todos os itens que contituem as portas dianteiras e traseiras, tampa da mala e capô.

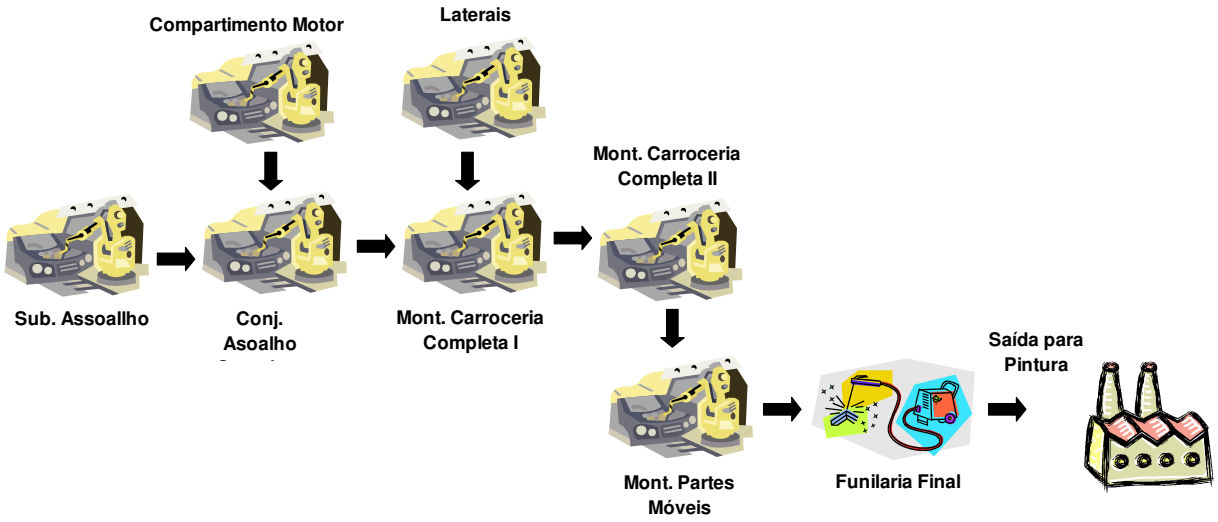

Figura 4.2: Representação hierárquica da fábrica de carrocerias no nível de linhas de produção.

No esquema acima, podemos representar a relação entre a fábrica de carroceria e a pintura por sua última atividade, ou seja, a Funilaria Final. Neste contexto, o gráfico da Figura 4.3 apresenta os dados referentes ao status da Funilaria Final no período de Maio a Dez de 2004:

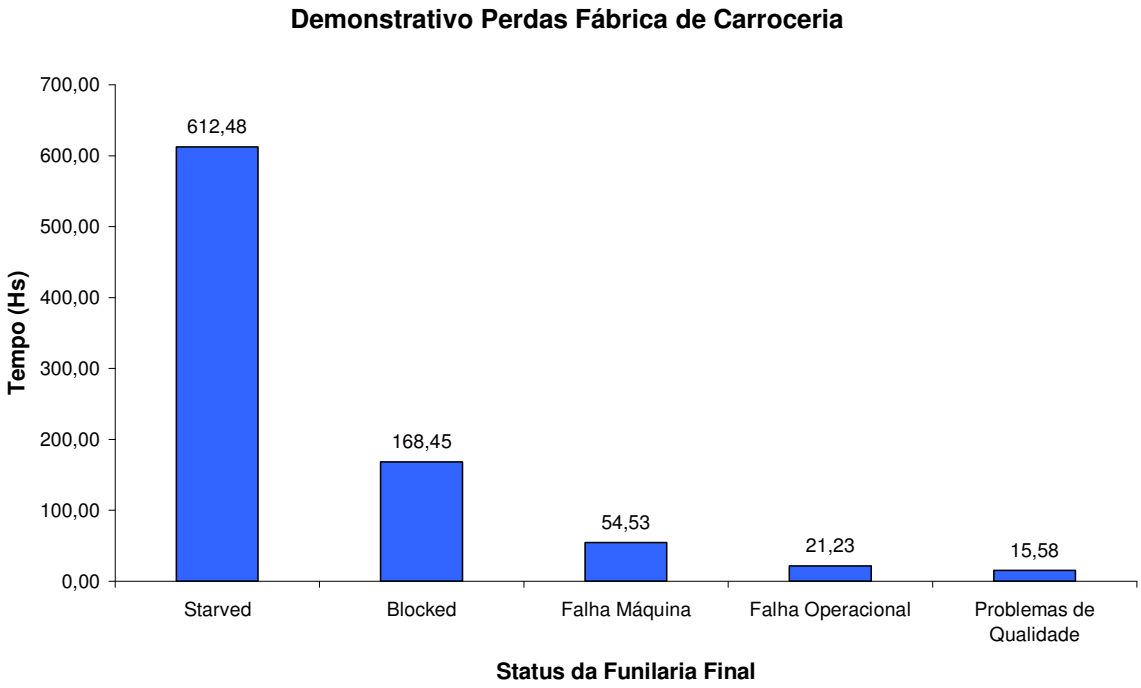

Figura 4.3: Demonstrativo de perdas da fábrica de carroceria, período maio a dezembro 2004.

Desta forma, com base nos dados apresentados acima, as perdas na fábrica de carrocerias estão distribuídas da seguinte forma:

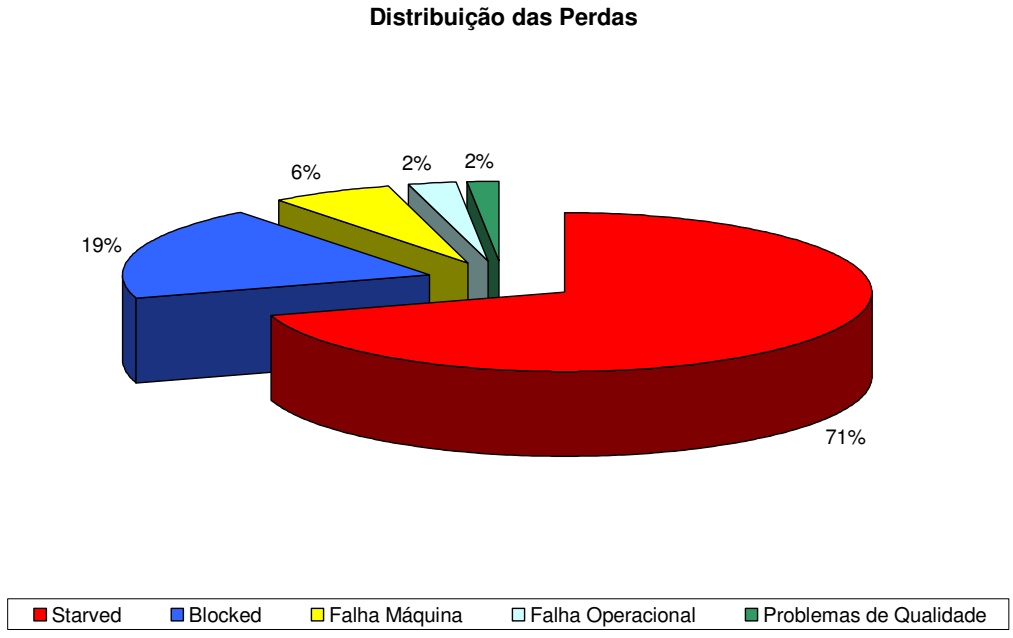

Figura 4.4: Demonstrativo da distribuição de perdas da fábrica de carroceria, período Maio a Dezembro 2004.

Pelos resultados acima, as perdas pelo status "*starved*", isto é, à espera de unidades para que sejam processadas pela linha de produção da Funilaria Final equivale a 612,48 horas (34.345 carros) ou 71% das perdas ocorridas, constituindo-se no mais significativo e por isso o foco central das observações. O status "*starved*" significa que a Funilaria Final ficou à espera de unidades a serem processadas e entregues à Pintura. Assim, neste trabalho a análise de performance é focalizada nas linhas de produção até a Linha de Montagem da Carroceria Completa, uma vez que as partes móveis são entregues em *batch* e, historicamente não têm afetado na performance da Funilaria Final.

A Tabela 4.1, mostra os dados de perdas de produção nas demais linhas que constituem a fábrica de carrocerias no período de maio a dezembro de 2004.

Tabela 4.1: Demonstrativo de perdas nas linhas de produção da fábrica de carrocerias, período de maio a dezembro de 2004.

Tipo da Falha	Perdas por Linha de Produção (Hs)					
	Sub Assoalho	Comp. do Motor	Conj Assoalho	Laterais	Mont Carroceria I	Mont. Carroceria II
Blocked	30,93	37,77	316,52	64,33	365,82	233,47
Falha Máquina	97,75	56,95	43,28	90,50	46,47	37,83
Falha Operacional	323,12	585,73	519,70	270,33	253,60	354,20
Problemas de Qualidade	8,35	2,55	2,05	23,68	0,02	1,20
Starved	374,58	605,23	530,05	306,12	241,60	347,08

4.2 Simulação do Estudo de Caso na Indústria Automobilística

O estudo de caso apresentado neste trabalho concentra-se na modelagem e análise de performance de uma fábrica de carrocerias através da Rede de Petri p-t-Temporizada Modificada.

Nesta modelagem preferiu-se pelo desenvolvimento de um macro-modelo do sistema representado no nível de linhas de produção, nas quais representam um conjunto de processos de uma determinada função, e que também representa a rede de Petri refinada do sistema.

Conforme mostrado anteriormente, a fábrica de carrocerias compreende as seguintes linhas de produção ou unidades produtivas:

- Partes Inferiores: onde são fabricadas e montadas todas as peças necessárias à fabricação do assoalho completo;
- Partes Superiores: onde são fabricadas e montadas todas as peças necessárias à fabricação das laterais;

- Montagem da Carroceria: onde são fabricadas e montadas todas as peças necessárias à união do conjunto assoalho, laterais e teto.
- Partes Móveis: onde são fabricadas e montadas todas as portas dianteiras, traseiras, capô, etc..

Dentro destas linhas de produção existem células de produção específicas ao fluxo do processo produtivo do sistema.

Neste contexto, a modelagem do sistema é apresentada abaixo:

a) A rede apresenta 15 lugares e, representam os seguintes processos:
   - P1 representa todos os subconjuntos necessários para a produção do conjunto assoalho completo da carroceria, ou seja, a existência de assoalho traseiro, assoalho dianteiro, tala de sustentação lateral, suporte do motor, pára-choque traseiro, conjunto do compartimento do motor, reforço estrutural do assoalho completo, suporte painel de instrumentos e suporte da lateral;
   - P2 representa a montagem geométrica do assoalho completo;
   - P3 representa o *buffer* intermediário entre a montagem geométrica do assoalho e o reforço estrutural do assoalho completo;
   - P4 representa o reforço estrutural do assoalho completo;
   - P5 representa o *buffer* intermediário de conjuntos de assoalhos completos;
   - P6 representa a montagem geométrica da carroceria;
   - P7 representa o *buffer* intermediário entre a montagem geométrica da carroceria e o reforço estrutural da carroceria completa;
   - P8 representa o reforço estrutural da carroceria completa;
   - P9 representa o buffer intermediário de carrocerias completas antes dos processos de preparação para a entrega das carrocerias às linhas de montagem das partes móveis e funilaria final;

- P10 representa as linhas de montagem das partes móveis e funilaria final;
- P11 representa todos os subconjuntos necessários para a produção de laterais completas, ou seja, caixas de rodas, coluna D, coluna C, coluna B, coluna A, painel interno das laterais dir/esq, painel lateral externo dir/esq e os subconjuntos internos e externos montados;
- P12 representa a linha automática de fabricação de laterais completas;
- P13 representa a célula de produção de saída de laterais, realizando o depósito em um elevador;
- P14 representa as laterais completas no sistema aéreo de transporte;
- P15 representa as laterais completas já transportadas para serem fornecidas ao processo de montagem geométrica da carroceria completa, ou seja, um *buffer* intermediário de laterais;

b) A rede apresenta 14 transições que representam as seguintes atividades ou eventos:

- T1 representa os processos de fabricação dos subconjuntos necessários à fabricação de um assoalho completo;
- T2 representa a fabricação geométrica de um assoalho completo;
- T3 representa o transporte de assoalhos para o processo de reforço estrutural;
- T4 representa o processo de reforço estrutural do assoalho completo;
- T5 representa o transporte sincronizado de conjuntos assoalhos completos e conjunto de laterais completas para o processo de montagem geométrica da carroceria;
- T6 representa o processo de montagem geométrica da carroceria completa;
- T7 representa o transporte de carrocerias completas para o processo de reforço estrutural da carroceria completa;

- T8 representa o processo de reforço estrutural da carroceria completa;
- T9 representa o transporte das carrocerias completas para as linhas de montagem de portas e funilaria final;
- T10 representa a renovação de matéria-prima devido ao próprio ciclo produtivo, bem como, utilizada para eliminar a existência de lugares fontes (*source*) ou drenos (*sinks*) no modelo do sistema;
- T11 representa os processos de fabricação dos subconjuntos de laterais;
- T12 representa a fabricação da lateral completa na linha automática;
- T13 representa o depósito de laterais completas no sistema de transporte aéreo;
- T14 representa o translado de laterais completas da linha de produção de laterais até a linha de produção da montagem completas de carrocerias através do sistema de transporte aéreo;

Nesta modelagem o tempo do respectivo processo é representado pelo tempo da célula de processo gargalo, isto é, o tempo do processo que limita o subsistema em consideração. Estes tempos de ciclo dos processos foram cedidos pela montadora do sistema em análise. A Figura 4.5 abaixo mostra o modelo do sistema estudado neste trabalho.

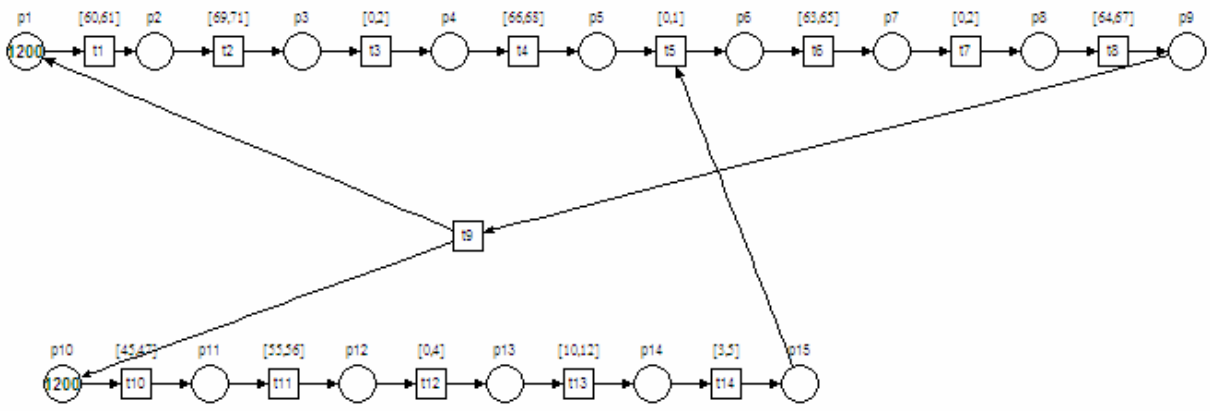

Figura 4.5: Modelo do estudo de caso em estudo.

Para o modelo acima, a matriz de incidência, vetor de capacidade e o vetor de marcação inicial são apresentadas, respectivamente, pelas equações (34), (35) e (36), abaixo:

$$A = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \quad (34)$$

$$K = \begin{bmatrix} 3000 \\ 1 \\ 19 \\ 1 \\ 42 \\ 1 \\ 39 \\ 1 \\ 3 \\ 3000 \\ 3000 \\ 23 \\ 1 \\ 1 \\ 86 \end{bmatrix} \quad (35)$$

$$M_0 = \begin{bmatrix} 3000 \\ 0 \\ 0 \\ 0 \\ 42 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 3000 \\ 0 \\ 0 \\ 0 \\ 86 \end{bmatrix} \quad (36)$$

Os tempos referentes às duas funções associadas a cada lugar relacionado ao equipamento, isto é, $I_{pi} = [MTBF_i, MTTR_i]$, é apresentado na Tabela 4.2 abaixo, onde são apresentados os valores teóricos de projeto e os valores mensurados do sistema em sua realidade. Os tempos reais do sistema foram cedidos pelo setor de Manutenção do sistema em estudo, onde se verifica um grande foco nas linhas principais, mas por outro lado uma ausência ou um acompanhamento deficiente na medição de desempenho dos sistemas auxiliares. Na simulação do modelo, os casos onde os tempos não são mensurados foram associados à não ocorrência de falhas. Foram feitas 07 simulações neste estudo de caso, com os seguintes objetivos:

1. Simular o sistema em suas condições ideais, onde não se apresenta atrasos nos transportes entre as células de produção e não há a ocorrências de falhas. Porém todos os buffers estão inicialmente vazios, com o objetivo de se verificar a influência dos mesmos no objetivo final em termos da quantidade de itens produzidos;
2. Simular a influência dos atrasos dos transportes em relação ao objetivo final. Mais precisamente, analisar a influência da comunicação CAN Bus (Control Área Network) entre os sensores, controladores e atuadores nas redes de comunicação atualmente utilizadas, com o intituito de se verificar a viabilidade técnica e econômica em projetos de melhorias das mesmas;

3. Simular qual a máxima quantidade de itens obtidos poder-se-ia obter com o sistema funcionando nas condições de projeto, isto é, identificar a máxima capacidade do sistema, para orientar nos cálculos dos indicadores de rendimento operacional global;
4. Simular, qual a máxima quantidade de itens obtidos que se pode obter com os dados reais de ocorrência de falhas, levando-se também em consideração as influências dos atrasos de transporte e, das quantidades dos buffers, para se comparar com os dados e informações atualmente registradas, de modo a identificar outras fontes de perdas de produção atualmente não identificadas ou não registradas e contabilizadas.

Tabela 4.2: Dados de confiabilidade e manutenabilidade do projeto e medidos pelo setor de Manutenção

	Projeto		Real	
LUGAR	MTBF(horas)	MTTR(min)	MTBF(horas)	MTTR(min)
P1	∞	0	Não medido	Não medido
P2	1.5	5	2.22	5.16
P3	∞	0	Não medido	Não medido
P4	1.5	5	8.0	8
P5	∞	0	Não medido	Não medido
P6	1.5	5	1.18	6.42
P7	∞	0	Não medido	Não medido
P8	1.5	5	1.84	4.32
P9	∞	0	Não medido	Não medido
P10	∞	0	Não medido	Não medido
P11	∞	0	Não medido	Não medido
P12	1.5	5	0.82	6.8
P13	1.5	5	468	6
P14	∞	0	21.5	2.5
P15	∞	0	Não medido	Não medido

Para o estudo de caso foram realizadas simulações variando-se as características do sistema quanto ao atraso no tempo de transporte, situação inicial dos *buffers* e a confiabilidade dos subsistemas.

Os resultados das simulações foram analisados considerando-se as seguintes variáveis:

i. Quantidade de itens produzidos na saída do sistema;

ii. A situação dos *buffers* intermediários;
iii. A eficiência do sistema e dos subsistemas produtivos;
iv. Quantidade de itens com a produção iniciada, isto é, o início da cadeia produtiva;

A Tabela 4.3 abaixo mostra as considerações feitas no modelo para a simulação do sistema em relação às variáveis discutidas anteriormente. Em todas as simulações foi considerado o tempo de simulação equivalente a um dia completo de produção (21,5 horas de trabalho).

Tabela 4.3: Considerações realizadas no modelo para a execução das simulações do sistema

SIMULAÇÃO	TEMPO DE ATRASO NO TRANSPORTE	SITUAÇÃO INICIAL DOS BUFFERS	CONFIABILIDADE E MANUTENABILIDADE
#01	0 seg	Vazios	Sem falhas
#02	0-2 seg	Vazios	Sem falhas
#03	1-3 seg	Vazios	Sem falhas
#04	0 seg	Vazios	Dados de projeto
#05	0 seg	Vazios	Dados reais
#06	1-3 seg	Vazios	Dados reais
#07	1-3 seg	Lugares P5 e P15 na Cap. Máxima	Dados reais

A Figura 4.6 abaixo mostra a influência no resultado da saída do sistema considerando-se a existência do atraso nos tempos de transporte. É observado que atrasos de 0-3 segundos afetam na saída de 4 a 12 unidades por dia, que resultam em 1248 a 3744 unidades ao ano. A eficiência do sistema se reduz de 97,35% para 96,28%. Tal resultado nos mostra que é viável um investimento na revisão do projeto das redes de comunicação do chão-de-fábrica de modo a garantir a não existência de atrasos nos tempos de transporte.

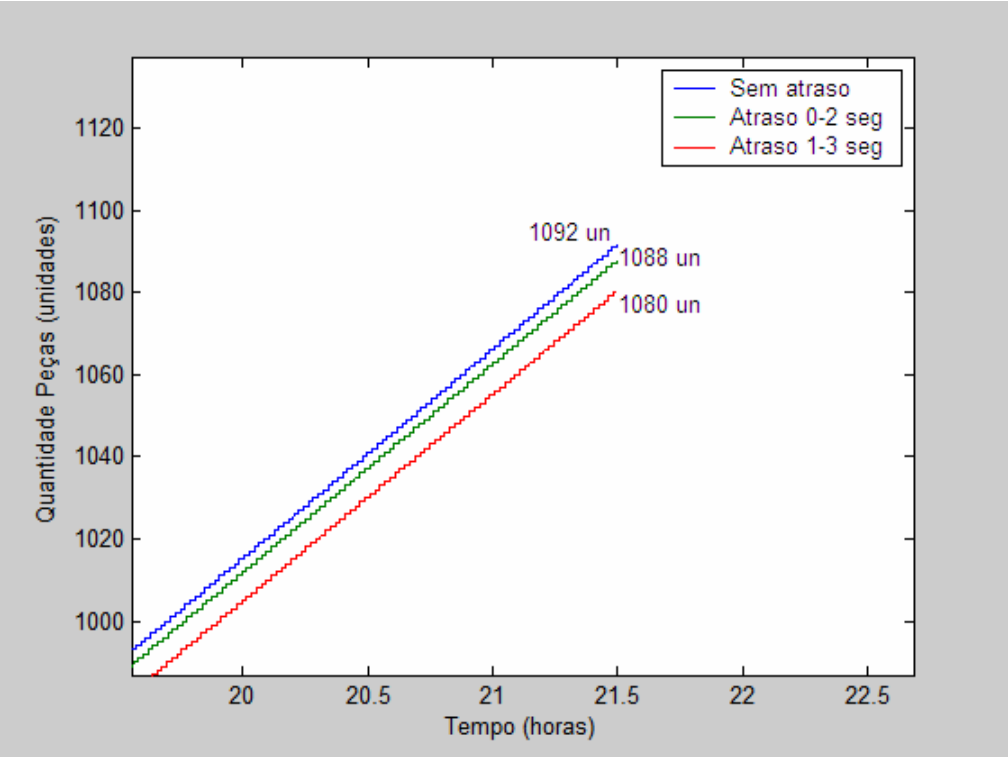

Figura 4.6: Influência dos atrasos nos tempos de transporte na quantidade de saída de unidades no final do sistema.

A Figura 4.7 abaixo mostra a influência no resultado da saída do sistema considerando-se a ocorrência de falhas no sistema, isto é, levando-se em consideração as características de confiabilidade e manutenabilidade dos equipamentos. Neste caso é apresentada a comparação do sistema com a confiabilidade máxima (sem falhas), os valores de confiabilidade do projeto, e os valores de confiabilidade reais obtidos. A eficiência do sistema com os valores de confiabilidade do projeto e os valores de confiabilidade reais são de 91,29% e 91,82%, respectivamente. Considerando-se a saída do sistema, a perda em relação ao sistema sem falhas varia de 62 – 68 unidades/dia.

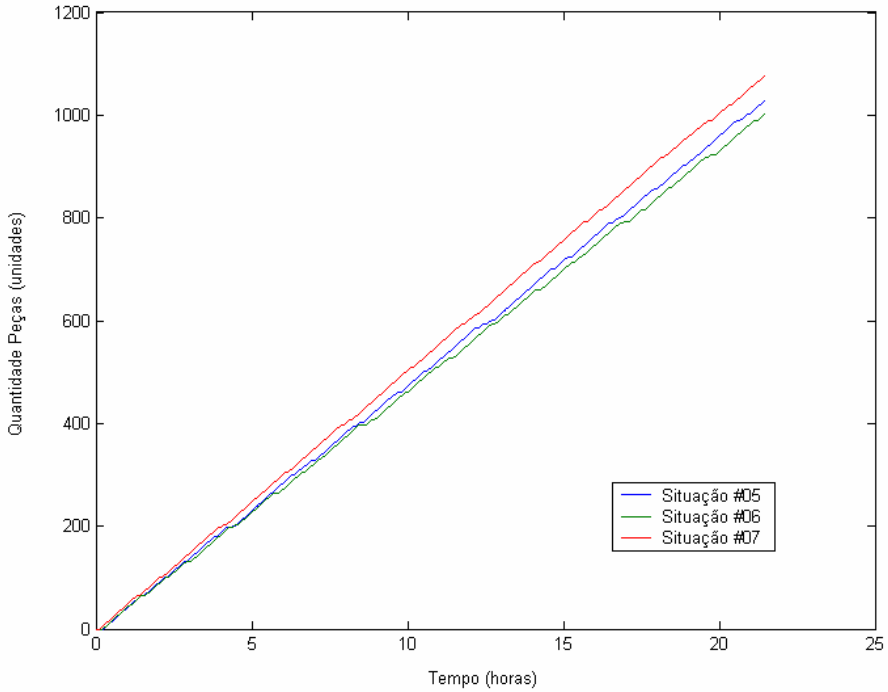

Figura 4.7: Influência da confiabilidade e manutenabilidade dos equipamentos do sistema na quantidade de saída de unidades no final do sistema.

O fato observado acima, referente a uma maior eficiência do sistema com os valores de confiabilidade reais em relação aos valores de confiabilidade de projeto, se devem a maior taxa de utilização do gargalo, que neste caso é o lugar P1 que representa a linha de produção de montagem geométrica do assoalho completo. O MTBF de projeto é de 1,5 horas, enquanto que o MTBF real é de 2,2 horas, confirmando a premissa de que quanto maior for a confiabilidade do sistema, maior será a eficiência do sistema (KUO & HUANG, 2000). A Figura 4.8 mostra a relação das taxas de utilização entre as 03 situações do sistema apresentado acima.

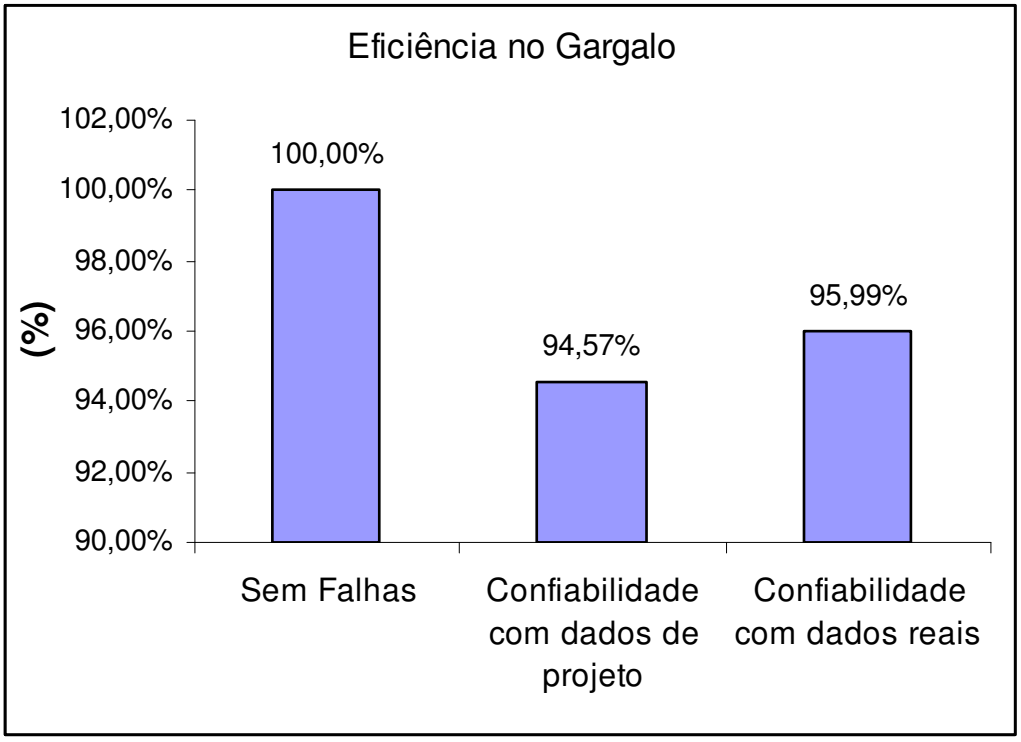

Figura 4.8: Influência da confiabilidade e manutenabilidade dos equipamentos do sistema na Taxa de Utilização.

A Figura 4.8 também nos mostra que a taxa de utilização do gargalo é de 100%, entretanto, a eficiência do sistema na sua saída é de 97,35%. Esta situação ocorre devido ao sistema estar todo vazio inicialmente, isto é, os *buffers* estão todos vazios e, as taxas de utilização das linhas posteriores serão menores devido à espera de peças para o processamento. As taxas de utilização de cada linha são as seguintes:

1. Reforço estrutural do assoalho (P4) = 94,24%
2. Montagem geométrica da carroceria (P6) = 89,93%
3. Reforço estrutural da carroceria (P8) = 91,52%
4. Montagem das laterais (P12) = 97,30%

Assim, para maximizar a eficiência na saída, é o bastante adicionar no buffer de assoalhos (P5) o equivalente a 30 unidades. A Tabela 4.4 apresenta as taxas de utilização das linhas para cada situação simulada.

Tabela 4.4: Taxas de utilização obtidas em cada simulação do sistema

	Taxas de Utilização (%)				
Situação	**Lugar P2**	**Lugar P4**	**Lugar P6**	**Lugar P8**	**Lugar P12**
#01	100,00%	94,24%	89,93%	91,52%	97,30%
#02	98,97%	93,94%	89,65%	91,13%	98,89%
#03	97,56%	93,38%	88,95%	90,53%	98,89%
#04	94,57%	90,28%	85,54%	87,24%	94,40%
#05	95,99%	90,69%	86,11%	87,92%	87,72%
#06	94,03%	89,41%	83,55%	84,17%	87,72%
#07	94,07%	89,47%	89,68%	90,41%	87,72%

A Figura 4.9 mostra as variantes do caso real na eficiência da saída do sistema, considerando-se os atrasos nos tempos de transporte e a influência da utilização dos *buffers* intermediários para maximizar a saída do sistema.

Entretanto, considerando-se a situação inicial dos *buffers*, a melhor eficiência de saída do sistema é conseguida com os buffers referentes aos lugares P5 e P15, com 86 e 97 unidades, respectivamente. Porém, tais valores estão acima das suas capacidades, então a alternativa a ser utilizada passa a ser a melhoria da confiabilidade do sistema.

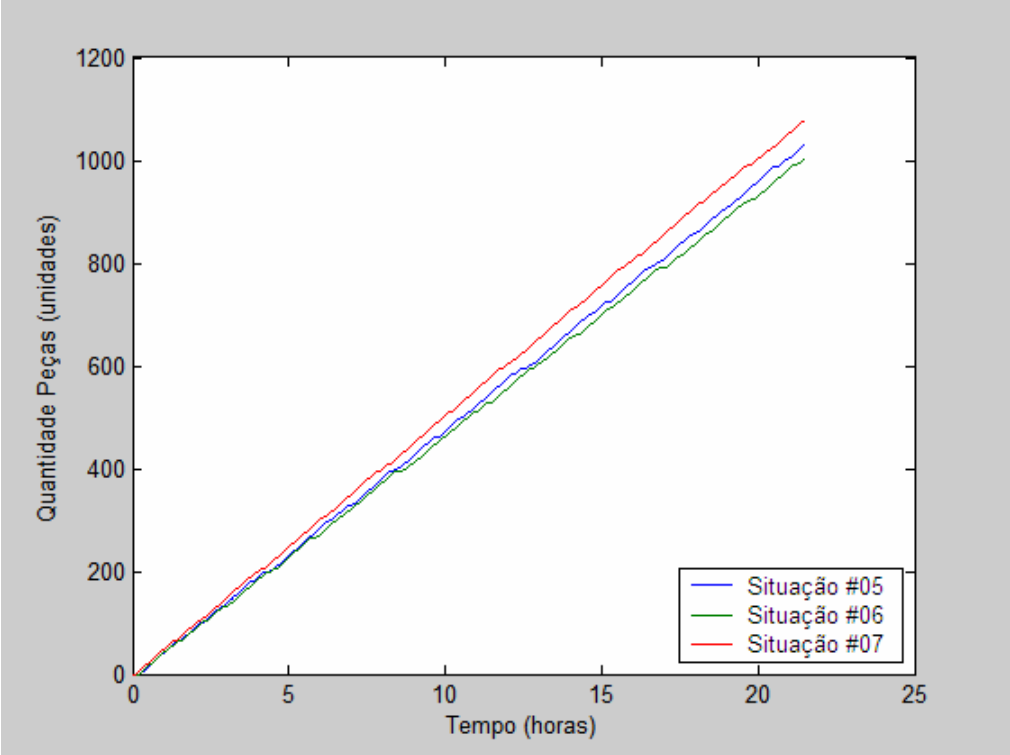

Figura 4.9: Influência do tempo de atraso no transporte e situação inicial dos buffers no sistema real.

A Figura 4.10 apresenta a influência das falhas na quantidade de unidades produzida pelo sistema.

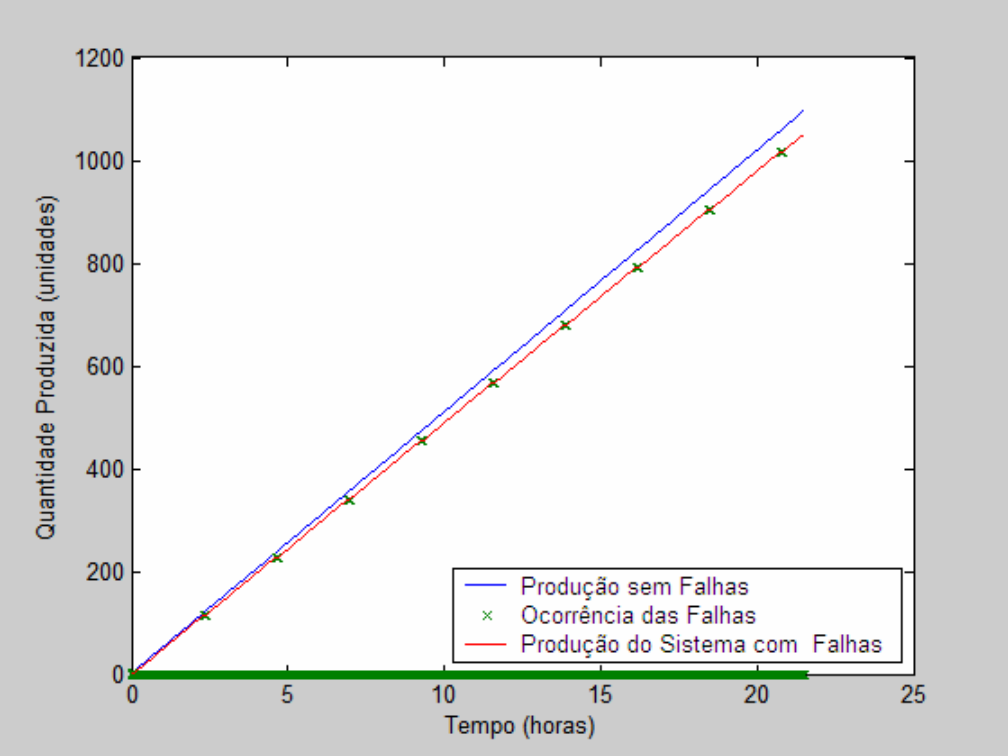

Figura 4.10: Influência das falhas na produção do sistema.

O programa desenvolvido para realizar a simulação no Matlab, considerando a abordagem de Rede de Petri p-t-Temporizada Modificada, para este estudo de caso é apresentado no Apêndice B.

5 Comentários Finais e Conclusões

Este trabalho introduz a utilização de Rede de Petri p-t-Temporizadas Modificada para a análise de performance de sistemas incluindo as características de confiabilidade e manutenabilidade de equipamentos. Esta nova rede permite a análise de performance utilizando uma linguagem de especificação de requisitos mais próxima da linguagem do chão-de-fábrica, sem a necessidade de usar a Redes de Petri Temporizada Estocástica. Esta abordagem facilita a avaliação dos requisitos de projeto referentes à confiabilidade dos processos e equipamentos para a obtenção da saída desejada sob o desempenho quantitativo do sistema.

A Rede de Petri p-t-Temporizada Modificada, como uma ferramenta gráfica, se torna um ponto comum de comunicação para projetistas e usuários, mesmo para sistemas mais complexos, pois preserva a simplicidade da representação gráfica do modelo, uma vez que não é necessário adicionar novos lugares e transições para a representar o sistema ou subsistema como disponível ou indisponível.

A modelagem através da Rede de Petri p-t-Temporizada Modificada também permite uma maior expressividade de modelagem, pois, propicia a utilização de mais de uma aplicação associada a um mesmo lugar desde que as mesmas representem eventos interdependentes.

Este trabalho também apresenta uma lógica linear a qual torna possível a modelagem e simulação dos sistemas utilizando Rede de Petri p-t Temporizada.

Neste trabalho foi introduzido um algoritmo polinomial para a reconfiguração da matriz de incidência do modelo inicial, através da automodificação surgida com a necessidade de representação da falha sem o acréscimo de elementos de rede.

O estudo também apresenta a influência da indisponibilidade e necessidade dos buffers intermediários na performance dos sistemas e, confirma a efetividade deste tipo de rede na análise de performance para os diferentes tipos de sistema do ponto de vista quantitativo, possibilitando a tomada de decisões com informações objetivas.

No próximo tópico são identificados alguns trabalhos futuros identificados durante a elaboração deste trabalho de forma a potencializar uma linha de pesquisa.

5.1 Trabalhos Futuros

Algumas sugestões para trabalhos futuros são:

1. Desenvolver um ambiente de simulação e modelagem adequada para a Rede de Petri p-t-Temporizada. Neste caso, temos duas opções viáveis ao nosso grupo de pesquisa:
    a. Implementação na rede Ghenesys ou,
    b. Implementação no software Tina em parceria com o LAAS (*Laboratoire d' Analyse et d'Architecture dês Systèmes*).
2. Desenvolver a fundamentação matemática necessária à análise de invariantes na Rede de Petri p-t-Temporizada.
    a. Desta forma, é possível a realização da análise de performance de forma algébrica, similar aos estudos realizados com as redes t-temporizadas e p-temporizadas existentes.
3. Aplicar o conceito da Redes de Petri p-t-Temporizada em sistemas supervisórios e de monitoração.
4. Desenvolver controladores e estratégias de controle geradas a partir dos modelos em Rede de Petri p-t-Temporizada.

6 Referências

ALUR, R., DILL, D. L., ***A theory of Timed Automata***, *Theoretical Computer Science*, Elsevier, vol. 126, pp. 183-235, 1994.

ARAÚJO, J. J., BECKER, L.B., PEREIRA, C.E., **Interface de Comunicação entre Ambiente de Modelagem Orientado a Objetos e Sistemas Supervisórios**, SBAI, 2001.

ASARIN, E., CASPI, P., MALER, O., ***A Kleene Theorem for Timed Automata***, *12th Annual IEEE Symposium on Logic in Computer Science*, Rússia, 1997.

BERTHOMIEU, B., DIAZ, M., ***Modeling and verification of time dependent systems using timePetri nets***, *IEEE Transactions on Software Engineering*, vol. 17, pp 259-273, Mar-1991.

BERTHOMIEU, B., RIBET, P.O., VERNADAT, F., ***The Tool TINA – Construction of abstract state spaces for Petri Nets and Time Petri Nets***, Int. J. Prod. Res, July 2004, Vol. 42, no. 14, pp. 2741-2756.

BEST, E., ***Design Methods Based on Nets***, *Lecture Notes in Computer Science* 424, Springer-Verlag, 1990.

BOUFAIED, A.,SUBIAS, A., COMBACAU, M., ***Chronicle modelling by Petri Nets for distributed detection of process failure***, IEEE SMC, 2002.

BUCCI, G., SASSOLI, L., VICARIO, E., ***Correctness Verification and Performance Analysis of Real-Time Systems Using Stochastic Preemptive***, *IEEE Transactions on Software Engineering*, vol. 31, no. 11, pp. 913-927, 2005.

CHANDRA, C., EVERSON, M., GRABIS, J., ***Evaluation of enterprise-level benefits of manufacturing flexibility***, Omega 33, pp. 17-31, 2005.

CHAUVET, F., HERMANN, J.W., PROTH, J., ***Optimization of Cyclic Production Systems: A Heuristic Approach***, *IEEE Transactions on Robotics and Automation*, Vol. 19, no. 01, Fevereiro 2003.

CHO, H, ***An Intelligent Workstation Controller for Computer Integrated Manufacturing***, Tese Doutorado, Texas *A&M University*, 1993.

COHEN, D. I. A., ***Introduction to Computer Theory***, *2nd edition*, John Wiley & Sons Inc., ISBN 0-471-13772-3, 1996.

CORTADELLA, J., LAVAGNO, L., ***Deriving Petri Nets from Finite Transition Systems***, *IEEE Transactions on Computer*, Vol. 47, no. 8, Agosto 1998.

DEL FOYO, P. M. G., Ghenesys: **Uma Rede Estendida Orientada a Objetos para Projeto de Sistemas Discretos**, Dissertação de Mestrado, USP, 2001.

DICESARE, F., HARHALAKIS, G., Proth, J.M., Silva, M., Vernadat, F.B., ***Practice in Petri Nets in Manufacturing***, Chapman & Hall, Londres, ISBN 0 412 41230 6.

DI FEBRARO, A., ***Optimization of Manufacturing Systems Modeled by Timed Petri Nets***, *Proceedings of Sixth International Workshop on Discrete Event Systems*, IEEE, 2002.

DROSTE, M., SHORTT, R.M., ***From Petri Nets to Automata with Concurrency***, *Applied Categorical Structures 10*, pags 173-191, Kluwer *Academic Publishers*, 2002.

GHALLAB, M., NAU, D., TRAVERSO, P., ***Automated Planning Theory and Practice***, Elsevier Inc., 2004, ISBN 1-55860-856-7.

GIUA, A.S., BASILE, F., ***Observer-based state-feedback control of timed Petri Nets with deadlock recovery***. IEEE Transactions on Automatic Control, pp. 17-29, vol. 49, ISSN 0018-9286, 2004.

GIULIA, A., DiCESARE, F., ***Supervisory Design Using Petri Nets***, *Proceedings of the 30th Conference on Decision and Control*, pp. 92-97, Brighton, Inglaterra, 1991.

HAAR, S., KAISER, L., SIMONOT-LION, F., TOUSSAINT, J., ***Equivalence of Timed Stated Machines and safe TPN***, *Proceedings of the 6th International Workshop on Discrete Event Systems*, IEEE, 2002.

HENNEMANN, F.A., RABELO, R. J., SANTOS, J.V.C, CURY, J.E.R, **Sistema de Apoio à Decisão Híbrido Utilizando Redes de Petri**, Simulação e Sistemas Especialistas, CBA, 2004.

HOPCROFT, J., MOTWANI, R., ***Introduction to Automata Theory, Languages and Computability***, Addison-Wesley Longman Publishing Co, ISBN:0201441241, 2nd edition, 2000.

JAIN, S., FOLEY, W., ***Impact of Interruptions on Schedule Execution in Flexible Manufacturing Systems***, The International Journal of Flexible Manufacturing Systems, n. 14, pp. 319-344, 2002.

JENG, M., XIE, X., HUNG, W., ***Markovian Timed Petri Nets for Performance Analysis of Semiconductor Manufacturing Systems***, IEEE Transactions on Systems, Man and Cybernetics, part B: *Cybernetics*, vol. 30, no. 5, October-2000.

JENSEN, K., ***Coloured Petri Nets: Basic concepts, analysis methods and pratical use***. *EATCS monographs on Theoretical Computer Science*, Springer-Verlag, Berlim, 1996.

JULIA, S., VALETTE, R., ***Real Time Scheduling of Batch Systems***, *Simulation Practice and Theory*, n. 8, pp. 307-319, 2000.

JUNQUEIRA, F. **Modelagem de Sistemas Flexíveis de Movimentação de Materiais através de Redes de Petri Interpretadas**. Tese de Mestrado, USP, 2001.

KOBAYASHI, H., ***Modeling and Analysis – An Introduction to System Performance Evaluation Methodology***, Addison-Wesley, 1978.

KUO, C., HUANG, H., ***Failure Modeling and Process Monitoring for Flexible Manufacturing Systems Using Colored Timed Petri Nets***, *IEEE Transactions on Robotics and Automation*, Vol. 16, n. 3, 2000.

LAVENBERG, S., S., ***Computer Performance Modeling Handbook***, Academic Press, 1983.

LEE, D.Y, DICESARE, F. ***Integrated Scheduling of Flexible Manufacturing Systems Employing Automated Guided Vehicles***, *IEEE Transactions on Industrial Electronics*, Vol. 41, no.6, Dez 2004.

LEE, J., KORBAA, O., ***Modeling and scheduling of ratio-driven FMS using unfolding time Petri Nets***, *Computers & Industrials Engineer*, no. 46, pp. 639-953, 2004.

MAIA, J., MORANDIN Jr. O., KATO, E.R.R., ***Discrete-Event Simulation and Logistics Management: Using the PPSS to Evaluate Supply Scheduling Decisions***, CBA, 2004.

MALINOWSKI, F.C., **Otimização do Sequenciamento de Veículos numa Linha de Montagem Utilizando Algoritmos Genéticos**, SBAI, 2001.

MANFRED, D., SHORTT, R.M., ***From Petri Nets to Automata with Concurrency***, *Applied Categorical Structures 10*, Kluwer Academic Publishers, Netherlands, pp. 173-191, 2002.

MARTÍNEZ Riascos, L.A. **Metodologia para Detecção e Tratamento de Falhas em Sistemas de Manufatura através de Redes de Petri**. Tese de Doutorado, USP, Junho 2002.

MEVIUS, M., PIBERNIK, R., ***Process Management in Supply Chains – A New Petri Net Based Approach***, *Proceedings on System Science*, 2004.

MONTGOMERY, E., **Introdução aos Sistemas a Eventos Discretos e à Teoria de Controle Supervisório**, Alta Books, 2005.

MURATA, T., ***Petri Nets: properties, analysis, and applications***, Proc. IEEE pp. 541-580, 1989.

MUSCAT, A., FLEURY, A., **Indicadores de Qualidade e Produtividade na Indústria Brasileira**, Revista Indicadores da Qualidade, no. 2, pp. 82-107, 1993.

NAKAMOTO, F.Y., **Regras de Controle para Alocação de Recursos em Sistemas Produtivos com Processos Concorrentes**, SBAI, 2001.

NAKASHIMA, K., GUPTA, S.M., ***Performance Evaluation of a Supplier Management System with Stochastic Variability***, *Int. J. Manufacturing Technology and Management*, Vol. 5, no. 1-2, 2003.

RESTREPO, P. L. A., **Modelagem Orientada a Objetos de Sistemas a Eventos Discretos: Estudo de Caso na Síntese de Controle de Sistemas Prediais**, Dissertação de Mestrado, USP, 2004.

SAITOU. K., MALPATHAK, S., QVAM, H., ***Robust design of flexible manufacturing systems using Colored Petri nets and genetic algorithm***, *Journal of Intelligent Manufacturing*, 13, 339-351, 2002.

SANTOS FILHO, D.J., SILVA, J.R., MARUYAMA, N., MIYAGI, P.E., **Estruturação da Modelagem de Processos em Sistemas Produtivos**, SBAI, 2001.

SAYGIN, C., KILIC, S.E., ***Dissimilarity Maximization Method for Real Time Routing of Parts in Random FMS***, *The International Journal of Flexible Manufacturing Systems*, no. 16, pp. 169-182, 2004.

SIMÃO, J. M., SILVA, P.R.O., STADZISZ, P.C., KÜNZLE, L.A., **Arquitetura de Software de Controle Orientada a Regras e Agentes para Sistemas Automatizados de Manufatura**, SBAI, 2001.

STARKE, P. H., ***Some Properties of Timed Nets under the Earliest Firing Rule***, *European Workshop on Applications and Theory in Petri Nets*, pp. 418-432, 1990.

TSINARAKIS, G. J., TSOURVELOUDIS, N. C., VALAVANIS, K. P., ***Modeling, analysis, synthesis, and performance evaluation of multioperational production systems with hybrid timed Petri nets***, *IEEE Transactions on Automation Science and Engineering*, vol. 3, pp 29-46, Jan-2006.

UDDIN, S., NOR, M.K., SALAM, S., ***Integration technique for an expert system on to a real-time system***. *In Proceedings of the TENCON'2000*, 2000.

ZHOU, M.C., DICESARES, F., RUDOLPH, D.L., ***Design and Implementation of a Petri Net Based Supervisor for a Flexible Manufacturing System***, Automatica, Vol.28, n. 6,pp. 1199-108, 1992.

ZHOU, M.C., McDERMOTT, K., PATEL, P., A., ***Petri Nets Synthesis and Analysis of a Flexible Manufacturing System Cell***, *IEEE Transactions on Systems, Man and Cybernetics*, Vol.23, n. 2, 1993.

ZHOU, M.C., ZHURAWSKI, R., ***Petri Nets and Industrial Applications: A Tutorial***, *IEEE Transactions on Industrial Electronics*, Vol. 41, n. 6, 1994.

ZHU, Q., SHENG, W., XI, N., ***Max-Plus Algebra Model for On-line Task Scheduling of a Reconfigurable Manufacturing Work-cell***, *Proceedings IEE/RSJ International Conference on Intelligent Robots and Systems*, Japão, 2004.

ZUBEREK, W., M., KUBIAK, W., ***Throughput Analysis of Manufacturing Cells Using Timed Petri Nets***, IEEE, 0-7803-2129-4, 1994.

ZUBEREK, W., M., ***Optimal Schedules of Manufacturing Cells – Modeling and Analysis using Timed Petri Nets***, IEEE, 0-7803-3334-9, 1996.

APÊNDICE A – Programa (para o ambiente Matlab) utilizado para a simulação da Rede de Petri Temporizada Modificada no exemplo do sistema produtor-consumidor

```
% Estudo de Caso - Esquema Produtor/Consumidor
% Modelagem de Redes de Petri Temporizadas
% Metodo de Ramchandani
% Inclusao da Nova Proposta Incluindo-se o MTBF e o MTTR
% Dissertacao de Mestrado - USP/Unifacs
% Aluno: Rossini Santos
% Orientador: Prof. Dr. Jose Reinaldo Silva
% Data de atualizacao: 14/10/2005

% Equacao de Estado
% Mk1 = Mk  + At*vk
% Onde:
%   Mk = Marcacao Atual
%   At = Matriz de Incidencia
%   vk = Vetor de habilitacao ou disparo
%   Mk1 = Nova marcacao

clc ;
clear ;

% Configuracao do Modelo
% Matriz de Incidencia

 At0 = [-1 0 0 0 1 0 0 0 0 0];
 At1 = [1 -1 0 0 0 0 0 0 0 0];
 At2 = [0 1 -1 0 0 0 0 0 0 0];
 At3 = [0 0 1 -1 0 0 0 0 0 0];
 At4 = [0 0 0 1 -1 0 0 0 0 0];
 At5 = [-1 1 0 -1 1 0 -1 1 -1 1];
 At6 = [0 0 0 -1 0 0 0 0 0 1];
 At7 = [0 0 0 0 1 0 0 0 -1 0];
 At8 = [0 0 0 0 0 -1 0 0 0 1];
 At9 = [0 0 0 0 0 1 -1 0 0 0];
At10 = [0 0 0 0 0 0 1 -1 0 0];
At11 = [0 0 0 0 0 0 0 1 -1 0];
At12 = [0 0 0 0 0 0 0 0 1 -1];
At = [At0; At1; At2; At3; At4; At5; At6; At7; At8; At9; At10; At11; At12];

% Marcacao Inicial

Mk0 = [1; 0; 0; 0; 0; 1; 4; 3; 0; 0; 0; 1; 0];

% Vetor de Capacidade
K = [1; 1; 1; 1; 1; 1; 7; 7; 1; 1; 1; 1; 1];

% Vetor delta - Tempos associados a cada transicao: Modelo
% Ramchandani
delta = [29; 29; 57; 29; 29; 56; 27; 27; 27; 27];

% Vetor de Rer Disparo nao atualizado
vz = 0*ones(10,1);

% Definicao do vetor de tempo local (p/ cada transicao)
tti = 0*ones(10,1);

% Inicializacao da Marcacao
Mk = Mk0;

% Evolucao dos Eventos com base no tempo
% Tempo Producao = 0.5hs ou 1800s
Tempo_Producao = 1800;
```

```
Tf = Tempo_Producao;

% Tratamento da Confiabilidade do Processo

% Definicao do vetor de tempo para falha e reparo (p/ cada lugar)
t_falha = 0*ones(13,1);
t_reparo = 0*ones(13,1);

% definicao do vetor flag de falha e sua referencia
flag_falha = 0*ones(13,1);
flag_ref = 0*ones(13,1);

% MTBF e MTTR associado aos lugares
%MTBF = [Tf; Tf; 570; Tf; Tf; 560; Tf; Tf; 560; Tf; Tf; Tf; Tf];
%MTTR = [0; 0; 285; 0; 0; 280; 0; 0; 280; 0; 0; 0; 0];
MTBF = [Tf; Tf; 1800; Tf; Tf; 1800; Tf; Tf; 1800; Tf; Tf; Tf; Tf];
MTTR = [0; 0; 90; 0; 0; 90; 0; 0; 90; 0; 0; 0; 0];

% Inicializacao dos Tempos
Tempo = 0;
tur = 0; % Tempo de utilizacao do robo
tum1 = 0; % Tempo de utlizacao M1
tum2 = 0;  % Tempo de Utilizacao M2

% Contador de Eventos - inicializacao
cont_evento = 0;

while Tempo <= Tempo_Producao
cont_evento = cont_evento + 1;

% Calculo do Vetor de Habilitacao
% Metodo Proposto por Del foyo
        % Proposicao 01 - Verificacao da Capacidade
% Mi = Mk*I + A
I = ones(1,10);
Mi = Mk*I + At;
vk = ones(10,1);

% Verificacao Proposicao 01
% v <= vk <= K
for i = 1 : 10
                for j = 1 : 13
                   if (Mi(j,i) < 0) | (Mi(j,i) > K(j,1))
                      vk(i) = 0;
                   end
                end
end

% Proposicao 02 - Verificacao de Conflito e Contato
% Matriz de checagem de conflito/contato -->> Mc = Mk + At*vk
Mc = Mk + At*vk;
for i = 1 : 13
                if (Mc(i,1) < 0)
                   disp('Tem conflito presente');
                   disp('Usando regra de decisao pre-definida');
                   lugar_conflito = i;
                end
end

% Regra de Decisao paa Eliminacao de Conflitos
if lugar_conflito == 6
                if vk(1) == 1
                   vk(4) = 0;
                   vk(7) = 0;
                   vk(9) = 0;
                elseif vk(4) == 1
                   vk(7) = 0;
                   vk(9) = 0;
                elseif vk(7) == 1
                   vk(9) = 0;
                end
end
            lugar_conflito = 0;
      % Definicao de vetor de marcas para verificacao da falha
      Mkm = Mk;
      Mk_check = 1;
```

```
vkt = 0*ones(10,1);
vkt_check = 1;
while ((vkt_check == 1) && (Mk_check == 1))

   % Incremento do Tempo Total
   Tempo = Tempo + 1;

   % Incremento de Tempo para o evento Falha - Associado ao lugar
      for i = 1 : 13
         if Mk(i) > 0
            t_falha(i) = t_falha(i) + 1;
         end
      end

   % Verificacao se ocorrera o evento falha (tempo operacao maior que o MTBF)
      if Mk == Mkm
         for a = 1 : 13
            if ((t_falha(a) >= MTBF(a)) && (flag_falha(a) == 0))
               flag_falha(a) = 1;
               disp('Atingiu o MTBF, a falha ocorreu');
               Mkm(a) = Mkm(a) - 1;
            end
         end
      end
   % Incremento Implicito de Tempo - MTTR associado ao lugar
      if Mk == Mkm
         for f = 1 : 13
            if flag_falha(f) == 1
               t_reparo(f)= t_reparo(f) + 1;
            end
         end
      end
   %  Verificacao para o retorno das marcas apos o reparo
      if Mk == Mkm
         for f = 1 : 13
            if ((t_reparo(f) >= MTTR(f)) && (flag_falha(f) == 1))
               Mkm(f) = Mkm(f) + 1;
               t_falha(f) = 0;
               t_reparo(f) = 0;
               flag_falha(f) = 0;
            end
         end
      end
   % Checagem de Mudanca da Marcacao para Recalcular o Vetor
   % Habilitacao
      %if Mk ~= Mkm
         %Mk = Mkm;
         %continue;
         %end


   % Incremento de Tempo para Habilitacao das Transicoes
      if Mk == Mkm
         for t = 1 : 10
            if vk(t) == 1
               tti(t)= tti(t) + 1;
            end
         end
      end

% Verificacao para o disparo das transicoes
   if Mk == Mkm
      for t = 1 : 10
         if tti(t) >= delta(t)
            vkt(t) = 1;
         end
      end
   else
      Mk_check = 0;
   end
   if vz == vkt
      vkt_ckeck = 1;
   else
      vkt_check = 0;
   end
```

```
      % Calculo da Performance
         % Checagem de falha
         for f = 1:13
            T_falha = flag_falha(f) + 1;
         end

         % Calculo Utilizacao Robo
         if Mk(6,1) == 0  && T_falha == 1
            tur = tur + 1;
         end
         % Calculo Utilizacao M1
         if Mk(2,1) == 1 && T_falha == 1
            tum1 = tum1 + 1;
         end
         % Calculo Utilizacao M2
         if Mk(9,1) == 1 && T_falha == 1
            tum2 = tum2 + 1;
         end
      end

% Proximos eventos
% Calculo da proxima marcacao
if Mk == Mkm
Mk1 = Mk + At*vkt;
         Mk = Mk1;
         for no_lugar = 1: 13
   Mk_saida(no_lugar,cont_evento) = Mk1(no_lugar);
         end
            Mk_saida(14,cont_evento) = Tempo;
         % Comparacao dos vetores vk e vkt
         % Proposicao: evitar interrupcao do disparo ja efetuado
         %          : evitar um novo disparo

            for trans = 1 : 10
               if vk(trans) == vkt(trans)
                  tti(trans) = 0;
               end
            end
      else
         Mk = Mkm;
      end
      % Taxas de Utilizacao
      Rtur = tur/Tempo_Producao;
      Rtum1 = tum1/Tempo_Producao;
      Rtum2 = tum2/Tempo_Producao;

end
```

APÊNDICE B – Programa (para o ambiente Matlab) utilizado para a simulação da Rede de Petri Temporizada Modificada no estudo de caso na indústria automobilística.

```
% Estudo de Caso - Industria Automobilistica 01
% Modelagem de Redes de Petri Temporizadas
% Metodo de Ramchandani e Merlin
% Inclusao da Nova Proposta Incluindo-se o MTBF e o MTTR
% Dissertacao de Mestrado - USP/Unifacs
% Aluno: Rossini Santos
% Orientador: Prof. Dr. Jose Reinaldo Silva
% Data de atualizacao: 19/03/2007
% Revisao 21/03/07 - a)Algoritmo Merlin; b) Retirada de todas as fichas do lugar
%                    apos a falha; c) Correçao do sistema apos falha
%                    (retirada da transiçao)

% Equacao de Estado
% Mk1 = Mk  + At*vk
% Onde:
%   Mk = Marcacao Atual
%   At = Matriz de Incidencia
%   vk = Vetor de habilitacao ou disparo
%   Mk1 = Nova marcacao

clear
clear
clear
clc ;
clear ;
disp('Pressione Enter para Continuar Execuçao')
pause

% Configuracao do Modelo
% Matriz de Incidencia

 A0 = [-1 1 0 0 0 0 0 0 0 0 0 0 0 0 0];
 A1 = [0 -1 1 0 0 0 0 0 0 0 0 0 0 0 0];
 A2 = [0 0 -1 1 0 0 0 0 0 0 0 0 0 0 0];
 A3 = [0 0 0 -1 1 0 0 0 0 0 0 0 0 0 0];
 A4 = [0 0 0 0 -1 1 0 0 0 0 0 0 0 0 -1];
 A5 = [0 0 0 0 0 -1 1 0 0 0 0 0 0 0 0];
 A6 = [0 0 0 0 0 0 -1 1 0 0 0 0 0 0 0];
 A7 = [0 0 0 0 0 0 0 -1 1 0 0 0 0 0 0];
 A8 = [0 0 0 0 0 0 0 0 -1 1 0 0 0 0 0];
 A9 = [1 0 0 0 0 0 0 0 0 -1 1 0 0 0 0];
A10 = [0 0 0 0 0 0 0 0 0 0 -1 1 0 0 0];
A11 = [0 0 0 0 0 0 0 0 0 0 0 -1 1 0 0];
A12 = [0 0 0 0 0 0 0 0 0 0 0 0 -1 1 0];
A13 = [0 0 0 0 0 0 0 0 0 0 0 0 0 -1 1];

A = [A0; A1; A2; A3; A4; A5; A6; A7; A8; A9; A10; A11; A12; A13];
At = A';
At0 = At;

% Marcacao Inicial

Mk0 = [3000; 0; 0; 0; 42; 0; 0; 0; 0; 0; 3000; 0; 0; 0; 86]; % Com buffer
%Mk0 = [3000; 0; 0; 0; 0; 0; 0; 0; 0; 0; 3000; 0; 0; 0; 0]; %Sem buffer


% Vetor de Capacidade
K = [3000; 1; 19; 1; 42; 1; 39; 1; 3; 3000; 3000; 23; 1; 1; 86];

% Vetor delta - Tempos associados a cada transicao: Modelo
% Ramchandani (em segundos)
```

```
delta = [5; 62; 5; 61; 5; 63; 5; 62; 5; 78400; 15; 53; 15; 25];

% Vetor de Disparo nao atualizado
vz = 0*ones(14,1);
vpz = 0*ones(15,1);
vtz = 0*ones(15,1);

% Definicao do vetor de tempo local (p/ cada transicao)
tti = 0*ones(14,1);

% Inicializacao da Marcacao
Mk = Mk0;
Mk_antes_falha = 0*Mk;

% Evolucao dos Eventos com base no tempo
% Tempo Producao = 21.5hs ou 77400s
Tempo_Producao = 1.5*3600;
Tf = Tempo_Producao;

% Tratamento da Confiabilidade do Processo

% Definicao do vetor de tempo para falha e reparo (p/ cada lugar)
t_falha = randint(15,1)*3*3600;
t_reparo = 0*ones(15,1);

% definicao do vetor flag de falha e sua referencia
flag_falha = 0*ones(15,1);
flag_ref = 0*ones(15,1);

% MTBF e MTTR associado aos lugares
hr=3600;
% Simulaçao sem falha
%MTBF = Tf*ones(15,1)+ 36000*ones(15,1);
%MTTR = [0; 0; 285; 0; 0; 280; 0; 0; 280; 0; 0; 0; 0];
%Dados Coletados Set a Dez 2006
MTBF = [10*Tf; 2.22*hr; 10*Tf; 8*hr; 10*Tf; 1.18*hr; 10*Tf; 1.84*hr; 10*Tf; 10*Tf; 10*Tf; 0.82*hr; 6*4680; Tf; 10*Tf];
MTTR = [0; 5.16*60; 0; 4.25*60; 0; 6.42*60; 0; 4.32*60; 0; 0; 0; 6.8*60; 300; 1; 0];
% Dados de confiabilidade do projeto
%MTBF = [10*Tf; 1.5*hr; 10*Tf; 1.5*hr; 10*Tf; 1.5*hr; 10*Tf; 1.5*hr; 10*Tf; 10*Tf; 10*Tf; 1.5*hr; 6*4680; Tf; 10*Tf];
%MTTR = [0; 5*60; 0; 5*60; 0; 5*60; 0; 5*60; 0; 0; 0; 5*60; 300; 1; 0];

% Inicializacao dos Tempos
Tempo = 1;
Ttw = 0; % Tempo de utilizacao Tackweld
Tsw = 0; % Tempo de utlizacao Screweld
Tbs = 0;  % Tempo de Utilizacao Body Side
Tfr = 0; % Tempo de utilizaçao Framing
Tre = 0; % Tempo de utilizaçao Respot
Tfbs= 0; % Tempo de falta de Lateral
Tfub= 0; % Tempo de falta de cj. underbody
tempo_r = 0*ones(15,1); % Tempo de referencia para o reparo
tempo_fa = 0*ones(15,1); % Tempo de referencia da ocor. da falha
tempo_vk = 0*ones(14,1); % Tempo de referencia para disparo

% Contador de Eventos - inicializacao
cont_evento = 0;

Mkm = Mk;
cTX = 1;
flag_f = 0;

while Tempo <= Tempo_Producao
cont_evento = cont_evento + 1;
   Mkm = Mk;
   conflito = 0;
   lugar_conflito = 0*ones(15,1);
   % Verificaçao do Tempo de Simulaçao
   %Percentual de Processamento
   TP = (Tempo/Tf)*100;
   if TP == 0.1*cTX
      disp(TP);
      cTX = cTX + 1;

      break
   end
```

```
% Calculo do Vetor de Habilitacao para as transiçoes
% Metodo Proposto por Del foyo
   % Proposicao 01 - Verificacao da Capacidade
% Mi = Mk*I + A
I = ones(1,14);
Mi = Mk*I + At;
vk = ones(14,1);

% Verificacao Proposicao 01
% v <= vk <= K
for i = 1 : 14
      for j = 1 : 15
         if (Mi(j,i) < 0) | (Mi(j,i) > K(j,1))
            vk(i) = 0;
         end
      end
end
   % Verificaçao do Vetor de Habilitaçao apos modificacao da matriz de
   % incidencia - verificar regra para cada modelo
   if flag_f == 1
      for g = 1 : 14
         vmax = max(At(:,g)); % a notaçao ao lado seleciona toda a coluna
         vmin = min(At(:,g));
         if vmax > 0 && vmin < 0
            vk(g) = vk(g);
         else
            vk(g) = 0;
         end
      end
   end

   % Checagem para evitar mais de 01 disparo no mesmo instante de Tempo
   for df = 1 : 14
      if tempo_vk(df) == Tempo
         vk(df) = 0;
      end
   end

   %Checagem de deadlock
   if vk == 0
      disp('Ocorrencia de deadlock');
      disp(cont_evento);

   end

% Proposicao 02 - Verificacao de Conflito e Contato
% Matriz de checagem de conflito/contato -->> Mc = Mk + At*vk
Mc = Mk + At*vk;
for i = 1 : 15
      if (Mc(i,1) < 0)
         disp('Tem conflito/contato presente');
         lugar_conflito(i) = 1;
         disp(i);
         disp(cont_evento);

      end
end
   conflito=max(lugar_conflito);

% Regra de Decisao para Eliminacao de Conflitos/Contatos feita
% de acordo com o modelo em estudo
   if conflito == 1
      for e = 1 : 15
         if lugar_conflito(e) == 1
            loc1 = e;
            if loc1 == 1
               vk(10) = 0;
            else
               vk(e-1) = 0;
            end
            lugar_conflito(e) = 0;
            disp('Correçao Conflito/Contato');
            disp(e);
            disp(cont_evento);
         end
      end
```

```
    end
    % inicializaçao do vetor de transiçao modificado
    vkt = 0*ones(14,1);
    vkt_check = 1;

    % Definicao de vetor de marcas para verificacao da falha
    %Mkm = Mk;
    Mk_check = 1;

    % Inicializaçao do vetor de habilitaçao dos lugares

    vpk = 0*ones(15,1);
    vpkt = 0*ones(15,1);
    vpkt_check = 1;
    vtk = 0*ones(15,1);
    vtkt = 0*ones(15,1);
    vtkt_check = 1;


    % Verificaçao da habilitaçao dos lugares para falha
    for p = 1: 15
        if Mk(p) > 0 && tempo_fa(p) < Tempo
            vpk(p) = 1;
        end
    end
    % Verificaçao da habilitaçao dos lugares p/ reparo
    for r = 1: 15
        if Mk(r) == 0 && Mk_antes_falha(r) > 0 && tempo_r(r) < Tempo
            vtk(r) = 1;
        end
    end

% Vetor delta - Tempos associados a cada transicao: Modelo
% Merlin (em segundos) - Março 2007
tw = 69 + randint(1)*2; %56 + randint(1)*8;
    sw = 66 + randint(1)*2; %56 + randint(1)*7;
    fr = 63 + randint(1)*2; %58 + randint(1)*9;
    re = 64 + randint(1)*3; %56 + randint(1)*9;
    bs = 55 + randint(1)*1; %49 + randint(1)*4;
    %cm = 0 + randint(1)*0; % sem atraso no transporte
    %cm = 0 + randint(1)*2; % atraso no transporte 0 - 2s
    cm = 1 + randint(1)*2; % atraso no transporte 1 - 3s

delta = [cm; tw; cm; sw; cm-1; fr; cm; re; cm; 78400; cm+6; bs; cm+10; cm+3];
    %Verificaçao de valores negativos nos tempos de transiçao
    for d = 1 : 14
        if delta(d) < 0
            delta(d) = 0;
        end
    end

    % Verificaçao do Vetor de Habilitaçao do Lugar
    for i = 1 : 15
        if t_falha(i) >= MTBF(i) && vpk(i) == 1 && flag_falha(i) == 0
            vpkt(i) = 1;
            tempo_fa(i) = Tempo;
        end
    end

    % Verificacao se ocorrera o evento falha (tempo operacao maior que o MTBF)
    if Mk_check == 1
        for a = 1 : 15
            if ((vpkt(a) == 1) && (flag_falha(a) == 0) && Mk(a) > 0)
                flag_falha(a) = 1;
                disp('Atingiu o MTBF, a falha ocorreu no local: ');
                disp(a);
                disp(cont_evento);
                Mk_antes_falha(a) = Mkm(a);
                Mkm(a) = 0;
                Mk_check = 0;
                % Modificaçao da Matriz A apos a ocorrencia da
                % falha (valido tambem para redes nao
                % ordinarias)
                for b = 1 : 14
                    At(a,b) = 0;
                end
```

```
        end
      end
   end
   flag_f = max(flag_falha);
   %  Verificacao para o retorno das marcas apos o reparo
   if Mk_check == 1
      if flag_f == 1
         % Verificaçao do Vetor de Habilitaçao do Lugar p/reparo
         for re = 1 : 15
            if t_reparo(re) >= MTTR(re) && vtk(re) == 1 && flag_falha(re) == 1
               vtkt(re) = 1;
               tempo_r(re) = Tempo;
            end
         end

         for f = 1 : 15
            if ((vtkt(f) == 1) && (flag_falha(f) == 1))
               Mkm(f) = Mk_antes_falha(f);
               Mk_check = 0;
               t_falha(f) = 0;
               t_reparo(f) = 0;
               flag_falha(f) = 0;
               disp('correçao da falha no local');
               disp(f);
               disp(cont_evento);
               % Modificaçao da Matriz A apos a recuperaçao da
               % falha (valido tambem para redes nao
               % ordinarias)
               for c = 1 : 14
                  At(f,c) = At0(f,c);
               end

            end
         end
      end
   end

   % Verificacao para o disparo das transicoes
   if Mk_check == 1
      for t = 1 : 14
         if tti(t) >= delta(t) && vk(t) == 1
               vkt(t) = 1;
               tempo_vk(t) = Tempo;
         end
      end
   else
      Mk_check = 0;
   end
   % Verificaçao se existe alguma transicao habilitada
   % Transiçao habilitada
   if vz == vkt
      vkt_ckeck = 1;
   else
      vkt_check = 0;
   end
   % Lugar habilitado p/ falha
   if vpz == vpkt
      vpkt_ckeck = 1;
   else
      vpkt_check = 0;
   end
   % Lugar habilitado p/reparo
   if vtz == vtkt
      vtkt_ckeck = 1;
   else
      vtkt_check = 0;
   end
% Proximos eventos
if Mk_check == 1
% Calculo da proxima marcacao

Mk1 = Mk + At*vkt;
      Mkm1 = Mk; % Armazena matriz Mk passo anterior;
      vktm1 = vkt; % Armazena vetor de habilitaçao anterior;
      vkm1 = vk; % Armazena vetor de habilitaçao anterior;
```

```
    Mk = Mk1; % Evoluçao dos Passos;
    for no_lugar = 1: 15
Mk_saida(no_lugar,Tempo) = Mk1(no_lugar);
    end
    % Modificar o numero da Linha referente ao Armazenamento do
    % Tempo no Vetor de Saida
    Mk_saida(16,Tempo) = Tempo;


else
    Mkm1 = Mkm;
    vktm1 = vkt; % Armazena vetor de habilitaçao anterior;
    vkm1 = vk; % Armazena vetor de habilitaçao anterior;
    Mk = Mkm;
end
% Habilitaçao para o incremento dos tempos
if vkt_check == 1 && vpkt_check == 1 && vtkt_check == 1

    % Incremento do Tempo Global
    Tempo = Tempo + 1;
    % Incremento de Tempo para o evento Falha - Associado ao lugar
    for i = 1 : 15
        if Mk(i) > 0 && vpk(i) == 1
            t_falha(i) = t_falha(i) + 1;
        end
    end
    % Incremento Implicito de Tempo - MTTR associado ao lugar
    if Mk_check == 1
        if flag_f == 1
            for f = 1 : 15
                if ((flag_falha(f) == 1) && (vtk(f) == 1) && (Mk(f) == 0))
                    t_reparo(f)= t_reparo(f) + 1;
                end
            end
        end
    end

    % Comparacao dos vetores vk e vkt
    % Proposicao: evitar interrupcao do disparo ja efetuado
    %           : evitar um novo disparo

    for trans = 1 : 14
        if vk(trans) == vkt(trans)
            tti(trans) = 0;
        end
    end

    % Incremento de Tempo para Habilitacao das Transicoes
    if Mk_check == 1
        for t = 1 : 14
            if vk(t) == 1
                tti(t)= tti(t) + 1;
            end
        end
    end

    % Calculo da Performance

    % Verificaçao Buffer Zona 10
    Zona10(Tempo)=Mk(3,1);
    %Verificaçao Buffer Zona 20
    Zona20(Tempo)=Mk(5,1);
    %Verificaçao Buffer Zona 30
    Zona30(Tempo)=Mk(7,1);
    %Verificaçao Buffer Laterais;
    Laterais(Tempo)=Mk(15,1);
    % Quantidade de unidades produzidas
    Pc_out(Tempo)=Mk(10,1);
    % Calculo Utilizacao Tackweld
    if Mk(2,1) == 1  && Mk(3,1) < 19
        Ttw = Ttw + 1;
    end
    % Calculo Utilizacao Screw Weld
    if Mk(4,1) == 1  && Mk(5,1) < 42
        Tsw = Tsw + 1;
    end
```

```
        % Calculo Utilizacao Body Side
        if Mk(12,1) >= 1  && Mk(15,1) < 86
            Tbs = Tbs + 1;
        end
        % Calculo Utilizacao Framing
        if Mk(6,1) == 1  && Mk(7,1) < 39
            Tfr = Tfr + 1;
        end
        % Calculo Utilizacao Respot
        if Mk(8,1) >= 1  && Mk(9,1) < 3
            Tre = Tre + 1;
        end
        % Tempo Espera Framing por Lateral
        if Mk(5,1) >= 1  && Mk(7,1) < 39 && Mk(6,1)== 0 && Mk(15,1)== 0
            Tfbs = Tfbs + 1;
        end
        % Tempo Espera Framing por Cj. Underbody
        if Mk(5,1) >= 0  && Mk(7,1) < 39 && Mk(6,1)== 0 && Mk(15,1) > 0
            Tfub = Tfub + 1;
        end
        % Taxas de Utilizacao
        RTtw = Ttw/Tempo_Producao; %Tackweld
        RTsw = Tsw/Tempo_Producao; %Screw Weld
        RTbs = Tbs/Tempo_Producao; %Body Side
        RTfr = Tfr/Tempo_Producao; %Framing
        RTre = Tre/Tempo_Producao; %Respot
        RTfbs = Tfbs/Tempo_Producao; %Falta de Body Side
        RTfub = Tfub/Tempo_Producao; %Falta de Cj. Under Body
        % Armazenamento dos Tempos
        Time(Tempo) = Tempo/3600;
        % Armazenamento dos Locais de Falha
        Tw_Falha(Tempo) = flag_falha(2); %Tackweld
        Sw_Falha(Tempo) = flag_falha(4); %Screw weld
        Fr_Falha(Tempo) = flag_falha(6); %Framing
        Re_Falha(Tempo) = flag_falha(8); %Respot
        Bs_Falha(Tempo) = flag_falha(12); %Body Side
        %Eficiencia do Sistema
        Efic_Sis = (max(Pc_out))/(Tempo_Producao/69);


    end


end
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo