**GILVAN DE OLIVEIRA COSTA**

# Uma Plataforma Computacional de Suporte ao Ciclo de Desenvolvimento de Sistemas Automatizados de Manufatura.

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica do Paraná, como requisito parcial para obtenção do título de Mestre em Engenharia de Produção e Sistemas.

CURITIBA

2005

**GILVAN DE OLIVEIRA COSTA**

# Uma Plataforma Computacional de Suporte ao Ciclo de Desenvolvimento de Sistemas Automatizados de Manufatura.

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica do Paraná, como requisito parcial para obtenção do título de Mestre em Engenharia de Produção e Sistemas.

Área de Concentração: Automação e Controle de Sistemas

Orientador: Prof. Dr. Marco Antônio Busetti de Paula

Co-orientador: Prof. Dr. Eduardo A. Portela dos Santos

CURITIBA

2005

Costa, Gilvan de Oliveira
Uma Plataforma Computacional de Suporte ao Ciclo de Desenvolvimento de Sistemas Automatizados de Manufatura. Curitiba, 2006. 67p.

Dissertação de mestrado – Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Engenharia de Produção e Sistemas.

1.Teoria de Controle Supervisório 2.Sistemas a Eventos Discretos 3.Simulação 4.Ferramenta Case.

I.Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-Graduação em Engenharia de Produção e Sistemas.

## Agradecimentos

Primeiramente agradeço a Deus e a interseção de Nossa Senhora por todas as graças obtidas ao longo desse período.

Ao Prof. Busetti e ao Prof. Portela, pela confiança em mim e em meu trabalho, e pelas orientações, compreensão e incentivo, que possibilitam a execução desta projeto de pesquisa e a todos os outros professores pertencentes ao grupo PRODUTRONICA que diretamente ou indiretamente apoiaram meu trabalho.

A todos os funcionários do grupo PRODUTRONICA que tornara-se amigos que juntos foi possível tal realização.

À PUCPR, por me conceder esta oportunidade de realizar o mestrado.

A todos aqueles que, direta ou indiretamente, contribuíram para a realização deste trabalho.

## Sumário


LISTA DE FIGURAS.....V
LISTA DE TABELAS.....VI
LISTA DE ABREVIATURAS.....VII
RESUMO.....VIII
ABSTRACT.....IX
1 INTRODUÇÃO.....1
1.1. CONTEXTO.....1
1.2. PROPOSTA.....2
1.3. ESTRUTURA.....4
2 FUNDAMENTOS CONCEITUAIS.....6
2.1. DEFINIÇÃO DE SED.....6
2.2. LINGUAGENS.....7
2.2.1. REPRESENTAÇÃO DE SEDS ATRAVÉS DE LINGUAGENS..8
2.3. AUTOMATOS.....8
2.4. TEORIA DE CONTROLE SUPERVISÓRIO.....11
2.4.1. CONTROLABILIDADE.....12
2.4.2. SUPERVISORES.....13
2.4.3. CONTROLE MONOLÍTICO.....14
2.4.4. SÍNTESE DE SUPERVISORES MONOÍTICOS.....15
2.4.5. MODULARIDADE.....15
2.4.6. CONTROLE MODULAR LOCAL.....16
2.5. TÉCNICAS DE IMPLEMENTAÇÃO DO SISTEMA DE CONTROLE.....18
2.6. RESUMO DO CAPÍTULO.....22
3 ABORDAGEM DE DESENVOLVIMENTO.....23
3.1. ETAPA DE MODELAGEM.....25
3.2. ETAPA DE SÍNTESE.....27
3.3. ETAPA DE IMPLEMENTAÇÃO.....28
4 MODELAGEM DA FERRAMENTA CASE CODE GENERATOR.....30

4.1. RELACIONAMENTO ENTRE OS ELEMENTOS DO MÉTODO PROPOSTO.....30
4.1.1. AMBIENTE DE DESENVOLVIMENTO.....31
4.1.2. AMBIENTE DE VALIDAÇÃO.....32
4.1.3. AMBIENTE DE AUXILIO A IMPLEMENTAÇÃO.....32
4.2. DIAGRAMA DE ENTIDADE E RELACIONAMENTO.....32
4.3. DIAGRAMA DE CONTEXTO.....33
4.3.1. LISTA DE EVENTOS.....34
4.3.2. DFD DE RESPOSTA AO EVENTO 1.....35
4.3.3. DFD DE RESPOSTA AO EVENTO 2.....36
4.3.4. DFD DE RESPOSTA AO EVENTO 3.....37
4.3.5. DFD DE RESPOSTA AO EVENTO 4.....38
4.3.6. DFD DE RESPOSTA AO EVENTO 5.....39
4.3.7. DFD DE RESPOSTA AO EVENTO 6.....40
4.4. CÓDIGO DE IMPLEMENTAÇÃO DO SC.....41
4.5. CONSIDERAÇÕES SOBRE O PROCESSO DE DESENVOLVIMENTO.....41
5 EXEMPLO DE APLICAÇÃO DA FERRAMENTA CODE GENERATOR.....43
5.1. DESCRIÇÃO DA BANCADA DE TESTE (PROTÓTIPO DO SISTEMA DE MANUFATURA).....43
5.2. ETAPA DE MODELAGEM.....45
5.3. ESPECIFICAÇÕES E SUPERVISORES.....45
5.4. ETAPA DE SÍNTESE DOS SUPERVISORES MODULARES.....47
5.5. CADASTRO DO SISTEMA PRODUTO E SUPERVISORES MODULARES.....48
5.6. CARACTERÍSTICAS DA ESTRUTURA DE CONTROLE.....50
5.6.1. VALIDAÇÃO.....51
5.6.2. REALIZAÇÃO FÍSICA.....54
5.7. INTERFACE DE COMUNICAÇÃO.....59
5.8. RESUMO DO CAPÍTULO.....61
6 CONCLUSÃO E TRABALHOS FUTUROS.....62
REFERÊNCIAS BIBLIOGRÁFICAS.....64

## Lista de figuras


Figura 1.1 Estruturura do Documento.....5

Figura 2.1 (a) Autômato acessível; (b) Autômato não acessível.....10

Figura 2.1 (a) Autômato co-acessível; (b) Autômato não co-acessível.....10

Figura 2.3 Supervisor Monoítico.....15

Figura 2.4 Funcionamento dos Supervisores.....16

Figura 2.5 Arquitetura de controle proposta por QUEIROZ e CURY.....17

Figura 2.6 Sistema físico a ser controlado.....18

Figura 2.7 Autômato do sistema físico.....18

Figura 2.8 Especificações.....19

Figura 2.9 Supervisor Supc($G_{loc,1}$, $E_{loc,1}$) para a aplicação inicial.....20

Figura 2.10 Implementação dos supervisores.....20

Figura 2.11 Estrutura para sinais de desabilitação.....20

Figura 2.12 Estrutura para sinais de desabilitação de evolução de módulos.21

Figura 2.13 Estrutura geração dos sinais de habilitação, controle do módulo do sistema produto e controle dos estados.....21

Figura 2.14 Estrutura geração dos sinais de retorno, controle do módulo do sistema produto e controle dos estados.....22

Figura 3.1 Ciclo de desenvolvimento do sistema de controle para processos reconfiguráveis.....24

Figura 4.1 Diagrama de blocos dos módulos do sistema Code Generator.....31

Figura 4.2 Diagrama de Entidade e Relacionamento.....33

Figura 4.3 Diagrama de Contexto.....34

Figura 4.5 Estrutura da tela de Cadastro do Modelo da Planta.....36

Figura 4.6 Código do Autômato.....36

Figura 4.7 Estrutura da tela de Cadastro de Supervisores.....37

Figura 4.8 Código do Autômato.....37

Figura 4.9 Estrutura da tela de Configuração de sinais pela porta paralela....38

Figura 4.10 Estrutura da tela para Comunicar com Software de Simulação..39
Figura 4.11 Estrutura da tela para Comunicar com Dispositivos.....40
Figura 4.12 Estrutura da tela para Comunicar com Dispositivos (Micro-Controller).....41
Figura 5.1 Sistema MPS.....44
Figura 5.2 Modelagem das sub-plantas do sistema real.....45
Figura 5.3 Modelos das especificações para a aplicação inicial.....46
Figura 5.4 Exemplo de supervisores.....48
Figura 5.5 Codificação do autômato na linguagem reconhecida pelo GRAIL.....48
Figura 5.6 Tela do Sistema para inserção do Sistema Produto.....49
Figura 5.7 Tela do Sistema para inserção dos Supervisores.....50
Figura 5.8 Estrutura do SC sugerido para implementação.....51
Figura 5.9 Tela referente a realização do sistema de simulação.....52
Figura 5.10 Tela da Simulação do Sistema no Em-Plant.....53
Figura 5.11 Fluxograma do Sistema de Controle.....55
Figura 5.12 Exemplo de um circuito para interface de comunicação.....60
Figura 5.13 Simulação de sinais para a interface de comunicação.....60

## Lista de Tabelas

Tabela 4.1 Modelos utilizados no procedimento de síntese e supervisores locais resultantes.....19
Tabela 4.1 Lista de Eventos.....35
Tabela 5.1 Modelos utilizados no procedimento de síntese e supervisores locais resultantes.....47

# Lista de Abreviaturas

| | |
|---|---|
| **CLP** | Controlador Lógico Programável |
| **MPS** | Modular Production System |
| **SC** | Sistema de Controle |
| **SED** | Sistemas a Eventos Discretos |
| **TCS** | Teoria de Controle e Supervisório |

# RESUMO

Consiste do desenvolvimento de uma ferrameta case denominada Code Generator, capaz de auxiliar ao projetista de sistemas industriais automatizados, no desenvolvimento, validação e implementação de Sistemas de Controle.

A ferramenta está estruturada de acordo com a Teoria de Controle Supervisório e alguns aspecto encontrados dentro da manufatura virtual para a validação do sistema de controle obtido de acordo com o modelo matemático mencionado anteriormente.

Para a utilização e desenvolvimento de um sistema de controle é sugerido também aqui neste, uma metodologia de desenvolvimento comentada a posteriori, mas que consiste de três grandes fases de desenvolvimento: Modelagem e Síntese; Validação do Sistema de Controle através de simulação e por fim a implementação através da geração de códigos fontes, aqui para Micro-controlador ou Computador.

# ABSTRACT

This Master These is a development of a Case Tools called Code Generator, capable to assist the designer of automatized industrial system, for Development, Validation and Implementation of Control System. This tool is structualized in accordance with the Theory Control Supervisory and some aspect inside of the virtual manufacture for the validation of the control system.

To use and development of a control system it is also suggested here in this, a methodology of development, and this methodology consists of three great phase of development: Modeling and Synthesis: Validation of the System of Control through simulation and Implementation through the generation of codes sources, here for Micro-controller and Computer.

# 1
# Introdução

## 1.1. Contexto

O crescente aumento da complexidade dos sistemas automatizados, aliado a questões como segurança, confiabilidade, reusabilidade e distribuição fazem com que métodos formais para o projeto e implementação de sistemas de controle supervisório sejam desenvolvidos e aplicados. Na prática industrial corrente, os Controladores Lógicos Programáveis (CLP) são programados utilizando linguagens de baixo nível, tipicamente diagramas escada, o que resulta em programas muito extensos e de difícil compreensão e manipulação. O projeto do sistema de controle como um todo é realizado com base na experiência do projetista, e a verificação do mesmo é tipicamente realizada apenas através de experimentação ou simulação. Devido a complexidade dos programas de controle e dos sistemas de manufatura, tanto a verificação por experimentação quanto por simulação demandam muito tempo e recursos financeiros.

Fundamentos teóricos para a modelagem e controle de sistemas cuja dinâmica é determinada pela ocorrência de eventos discretos têm sido desenvolvidos. Os dois formalismoss mais comuns são as Redes de Petri e Máquinas de Estados Finitos; ambos permitem a verificação formal da correção do sistema de controle. Entretanto, apesar dos avanços significativos ocorridos nos últimos anos, estes métodos formais não têm sido empregados de forma significativa na indústria (VIEIRA, 2004).

A viabilidade do uso de metodologias de projeto para sistemas automatizados passa forçosamente pela disponibilidade de ferramentas computacionais que a implementem. Algumas destas estão hoje disponíveis porém, em geral, são desenvolvidas no meio acadêmico, e não têm as boas características de um produto, no que diz respeito a suas interfaces e capacidade de lidar com problemas

de porte. O desenvolvimento de ferramentas computacionais comerciais é de fundamental importância para a disseminação e aplicação de metodologias formais de projeto de sistemas automatizados de manufatura.

Dentre as ferramentas existentes para o apoio ao projeto de sistemas automatizados pode-se citar o TCT (WONHAM, 1999), DESCO (FABIAN e HELLGREN, 2000), UKDES (CHANDRA et al., 2002), GRAIL (RAYMOND e WOOD, 1996) e VER (BALEMI et al., 1993). A maioria destas implementa o modelo proposto por Ramadge e Wonham (1989), como o TCT, DESCO, UKDES e o VER. O GRAIL, originalmente concebido como uma ferramenta de manipulação de autômatos, tem agora algoritmos desenvolvidos pelo grupo de Automação da UFSC (CURY, 2001) que tratam todo o processo de síntese de controladores baseado na abordagem de Ramadge e Wonham (1989).

Outro ponto fundamental a ser considerado nas ferramentas citadas é a possibilidade de traduzir uma linguagem formal (por exemplo, autômatos) numa linguagem normalizada de programação de Controladores Lógico Programáveis (CLPs). Nesse aspecto pode-se citar o Supremica (AKESSON, 2002), onde a partir de modelos em autômatos, é possível gerar um código de acordo com a IEC 61131-3. Em relação a uma ferramenta que integra as etapas de desenvolvimento de um sistema automatizado, implementando o conceito de Engenharia Virtual, cita-se o VIR-ENG, resultado do projeto de pesquisa ESPRIT/IST (MOORE et al., 2003).

Nesse sentido, o presente trabalho tem por objetivo fundamental desenvolver uma ferramenta computacional de suporte ao ciclo de desenvolvimento de sistemas automatizados de manufatura. A partir de uma proposta de desenvolvimento de sistemas automatizados, a ferramenta auxilia o projetista na etapa de implementação do sistema de controle. Por sua vez, o modelo de implementação é baseado na Teoria de Controle Supervisório (RAMADGE e WONHAM, 1989) que tem seus fundamentos na teoria de Linguagens Formais e Autômatos (CARROL e LONG, 1989).

### 1.2. Proposta

Para o que foi exposto acima, a questão principal é:

*Como diminuir o tempo de desenvolvimento de projetos de sistemas de controle, usando Modelos Formais e Simulação de Sistemas?*

Para a problematização exposta, surgem algumas hipóteses relacionadas a seguir;

- Com a utilização de modelos formais, diminui o tempo de desenvolvimento de projetos de SC.
- Ferramentas de Simulação auxiliam no desenvolvimento de SC.

Este trabalho apresenta uma abordagem para o desenvolvimento de SC no contexto de sistemas automáticos de manufatura. A contribuição da abordagem proposta, caracterizada por um ciclo de desenvolvimento – modelagem, síntese e implementação – consiste em tratar o projeto de sistemas automatizados de manufatura com maior eficiência, eficácia e confiabilidade quando novas aplicações forem necessárias. Além de fornecer ao meio industrial uma ferramenta capaz auxiliar no desenvolvimento de SC seguindo as orientações descritas na abordagem proposta por esta pesquisa.

Em uma primeira etapa - modelagem, a abordagem utiliza a Teoria de Linguagens e Autômatos para a representação do sistema de manufatura (por exemplo, sistemas de manipulação, células de processamento, dentre outros) das especificações (por exemplo, requisitos operacionais, de segurança, de roteiros de produção, dentre outros). Em uma segunda etapa - síntese, supervisores modulares são concebidos a partir dos modelos do sistema de manufatura e das especificações, de acordo com a TCS. Em uma terceira etapa – implementação, ocorre a tradução dos supervisores modulares formais em algumas linguagem de acordo com necessidade e suporte da ferramenta desenvolvida, conforme estrutura de controle apresentada por Queiroz e Cury (2002). Para fins de validação, a estrutura de controle é inicialmente simulada, permitindo ao projetista a substituição ou modificação de modelos. Concluída esta etapa pode-se ter a realização física através da implementação em um determinado dispositivo de controle. A consolidação da etapa de implementação se dá no momento da realização completa da estrutura de controle concebida na etapa de síntese.

O ciclo de desenvolvimento é realizado através da integração de formalismos matemáticos (Teoria de Linguagens e Autômatos e TCS), de ferramenta computacional de síntese GRAIL (CURY, 2001), de bibliotecas de modelos de autômatos (de sub-sistemas e de especificações), e de técnica de simulação.

### 1.3. Estrutura

Está estruturado o presente documento em seis capítulos dispostos e organizados de acordo com as seguintes disposições:

- Capítulo 1, aqui apresentado, refere-se à introdução e apresentação dos objetivos.
- Capítulo 2 apresenta os fundamentos conceituais necessários para o desenvolvimento desta dissertação. Serão abordados conceitos referentes à metodologia baseada em Linguagem Formal e Autômatos, Controle Monolítico e Controle Modular Local, e tecnica de implementação do SC.
- Capítulo 3 traz a abordagem de desenvolvimento de projeto de SC aplicado a sistemas industriais.
- Capítulo 4 Trata da Modelagem da Ferramenta Code Generator, aqui mostra como foi concebida tal ferramenta case e algumas peculiaridades da mesma.
- Capítulo 5 Apresenta um Exemplo de Aplicação da Ferramenta Code Generator.
- Capítulo 6 apresenta-se a conclusão da presente dissertação

A figura 1.1 mostra a lógica e a estrutura do presente documento.

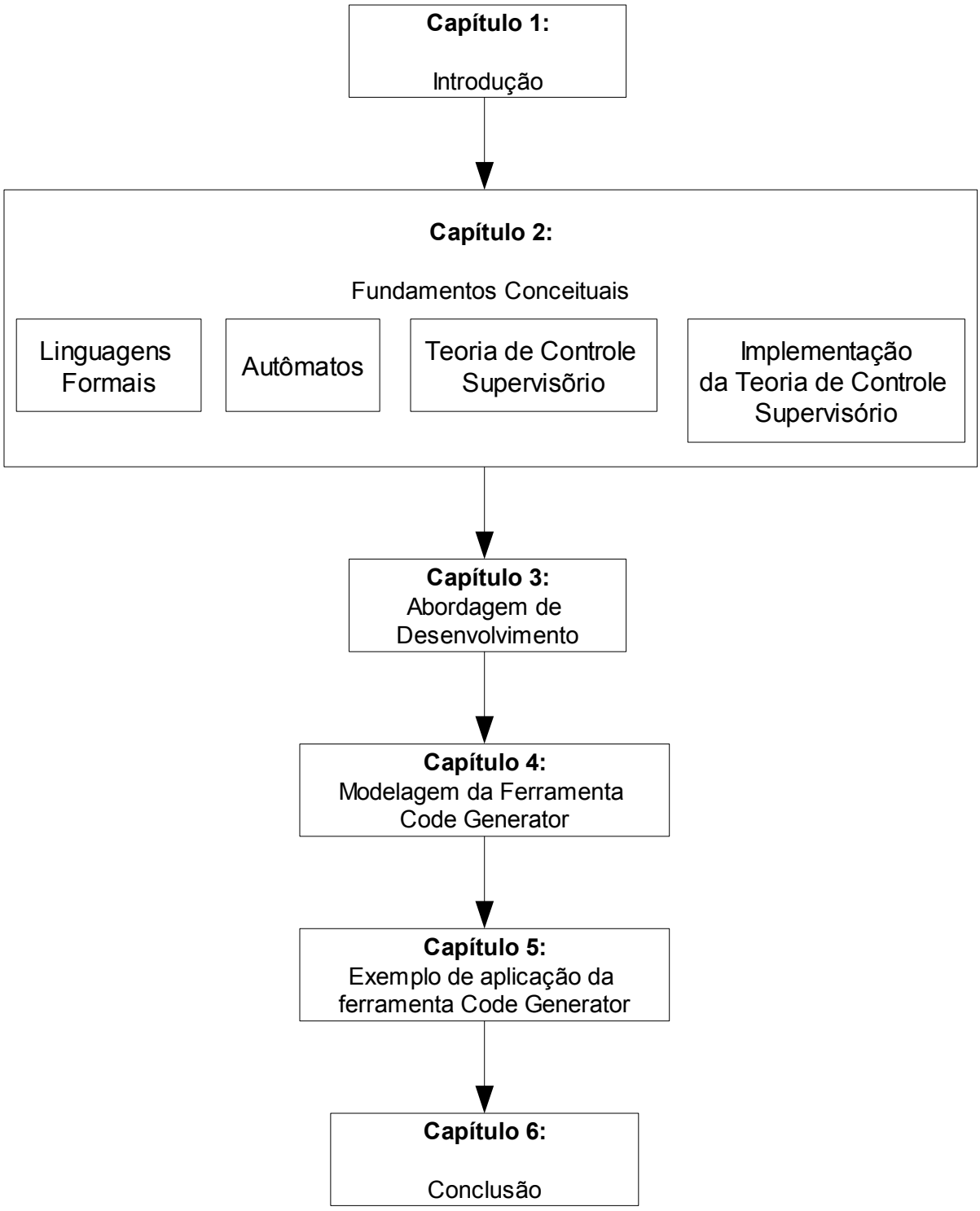


**Figura 1.1** Estrutura do documento.

# 2
# Fundamentos Conceituais

Este capítulo apresenta os fundamentos da teoria de controle de SEDs, introduzida por Ramadge e Wonham (1989). Inicialmente, é introduzida a classe de sistemas foco de estudo ao longo deste trabalho, denominada ‘Sistemas a Eventos Discretos’. Nas seções seguintes, conceitos fundamentias para a modelagem de SEDs sobre as quais se baseia e TCS, são apresentados. Além disso, uma extensão desta teoria, como o controle modular local, é discutida. Finalmente, é apresentado um modelo de implementação da TCS e do controle modular local.

## 2.1.Definição de SED

Define-se como SEDs todo conjunto de sistemas onde a mudança de estados, ou ainda a evolução do sistema, ocorre através de eventos discreto. A verificação e evolução do sistema ocorrerá através de estímulos denominados de eventos.

Conforme exposto anteriormente na seção 1.1, pode-se encontrar na literatura diversos modelos matemáticos a fim de solucionar problemas relacionados a projetos de SEDs tais como: Redes de Petri, Cadeias de Markov, Teoria das Filas, Processos Semi-Markovianos Generalizados e Simulação, Álgebra de Processos, Álgebra Max-Plus, Lógica Temporal, Autômatos e Teoria de Linguagens. Dentre os modelos relatado o presente documento visa explorar questões observadas pelo modelo proposto por Ramadge e Wonham (1989), onde o modelo mencionado e destacado refere-se a procedimento de síntese de controladores.

Assume-se que a representação matemática apresentada a seguir realiza a seguinte consideração:

i. A não ocorrência de dois ou mais eventos simultâneos em um mesmo instante de tempo.

ii. O modelo em questão não considera o instante de tempo da ocorrência de um determinado evento, mas sim a ordem de ocorrência dos mesmos.

## 2.2.Linguagens

Define-se uma linguagem $L$ definida sobre um alfabeto $\Sigma$ como sendo um conjunto de cadeias ou palavras formadas por símbolos pertencentes a esse alfabeto $\Sigma$ .

Define-se ainda que $\Sigma^*$ como conjunto de todas as cadeias finitas de elementos do conjunto $\Sigma$ , incluindo nele a cadeia vazia ε.

Conforme descrito por Cassandras e Lafortune (1999) existem algumas operações que podem ser realizadas com o conceito de linguagens, entre elas pode-se citar:

- **Concatenação:**

Considerando as linguagens $L_1$ e $L_2$ definidas como $L_1, L_2 \subseteq \Sigma$ , quando:

$$L_1 L_2 = \left\{ s \in \Sigma \mid (s = s_1 s_2), (s_1 \in L_1) e (s_2 \in L_2) \right\} .$$

Dessa forma a palavra formada pela nova linguagem $L_1 L_2$ pode ser escrita como a concatenação de duas outras linguagens a $L_1$ concatenada com $L_2$ .

- **Prefixo-Fechamento:**

Considerando a linguagem formada por $L \subseteq \Sigma$ onde:

$$\overline{L} = \left\{ s \in \Sigma \mid \exists t \in \Sigma (st \in L) \right\} .$$

$L \subseteq \overline{L}$ e quando $L = \overline{L}$ pode-se afirmar que a linguagem $L$ é prefixo-fechada.

- **Fechamento Kleene:**

Considerando a linguagem formada por $L \subseteq \Sigma$ pode-se dizer que fechamento Kleene da linguagem $L$ é dada por:

$$L^* = \{\varepsilon\} \cup L \cup LL \cup \dots .$$

Com base nas operações descritas por Cassandras e Lafortune (1999), pode-se concluir que em uma determinada palavra formada por $s = tuv$, e assumindo que $t,u,v \in \Sigma$, afirma-se que:

- $t$ é prefixo de $s$;
- $u$ é subcadeia de $s$;
- $v$ é sufixo de $s$;

### 2.2.1. Representação de SEDs através de linguagens

Na modelagem de um SED através de linguagens, associa-se os eventos passíveis de ocorrência a símbolos pertencentes ao $\Sigma$. O comportamento seqüencial de um SED pode ser representado através de um par de linguagens $L, L_m$ sobre um alfabeto $\Sigma$, onde se tem:

- $\Sigma$ é formado pelo conjunto de eventos do sistema que afetam a transição do estado do sistema;
- $L \subseteq \Sigma$ representa o comportamento gerado pelo sistema, ou ainda, o conjunto de todas as cadeias factíveis de ocorrerem pelo SED;
- $L_m \subseteq L$ representa o comportamento marcado do sistema, ou ainda, é um subconjunto das cadeias pertencentes a $L$, na qual representa o cumprimento de determinadas tarefas.

## 2.3. Automatos

Conforme descrito por Cassandras e Lafortune (1999) autômato é um dispositivo capaz de representar graficamente uma linguagem de acordo com algumas regras de concepção. A conexão entre linguagens e autômatos está nos

eventos de transição entre um estado e outro de um autômato, ou seja, os eventos de transição de um determinado autômato podem ser definidos de acordo com o embasamento matemático verificado na teoria de linguagem, sendo assim, uma linguagem pode ser gerada através de $G=(\chi,\Sigma,f,x_0,\chi_m)$ , onde:

- $\chi$ é um conjunto de estados;
- $\Sigma$ é definido pelo conjuntos dos eventos associado a transição de estados entre os autômatos;
- $f$ é a função de transição dos estados de acordo com $f:\chi$ x $\Sigma \rightarrow \chi$ ;
- $x_0$ é o estado inicial, logo $x_0 \in \chi$ ;
- $\chi_m$ é o conjunto de estados marcados, lembrando que $\chi_m \in \chi$ .

Verificado que um autômato pode representar graficamente uma linguagem, $L(G)$ é representada por todas as cadeias ou palavras que podem ser geradas no autômato partindo do estado inicial. A linguagem marcada $L_m(G)$ é formada por todas as cadeias ou palavras geradas partindo do estado inicial e que alcançam um determinado estado marcado. Sendo assim, conforme mencionado por Santos (2003), um SED pode ser modelado por um autômato $G$ , em que $L(G)$ é dito como comportamento gerado pelo sistema e $L_m(G)$ é o comportamento marcado.

Um autômato $G$ é dito acessível se todo $x \in \chi$ é acessível, formalmente tem-se que:

$$L(G)=(\chi,\Sigma,f,x_0,\chi_m)$$

Quando:

$$\chi=\left\{x \in \chi : \exists s \in \Sigma \left(f(x_0,s)=x\right)\right\}$$

Na figura 2.1 observa-se um exemplo da propriedade mencionada anteriormente a respeito de acessibilidade, onde é possível verificar que o autômato da figura 2.1 b não é acessível devido ao estado 2 não é alcançado partindo do estado inicial.


**Figura 2.1** (a) Autômato acessível; (b) Autômato não acessível.

Pode-se verificar também a co-acessibilidade dos estados de um autômato através de:

$$L(G)=(\chi,\Sigma,f,x_0,\chi_m)$$

Quando:

$$L(G)=\overline{L_m(G)}$$

Um autômato $G$ pode ser dito como co-acessível, ou ainda, como não bloqueante se for possível alcançar um estado marcado do sistema através de qualquer outro estado do mesmo através da ocorrência de alguns eventos pertencentes ao $\Sigma$, incluindo aqui a própria cadeia $\varepsilon$. Na figura 2.2 observa-se um exemplo da propriedade mencionada anteriormente a respeito de co-acessibilidade, onde é possível verificar que o autômato da figura 2.2 b não é co-acessível devido ao bloqueio do estado 2.


**Figura 2.2** (a) Autômato co-acessível; (b) Autômato não co-acessível com bloqueio.

Conforme descrito por Cassandras e Lafortune (1999), existem algumas operações possíveis com relação aos autômatos, será visto aqui apenas a composição síncrona, onde para CURY (2001) a modelagem de SEDs por autômatos pode ser abordada de duas formas: uma abordagem global e uma

abordagem local. Na abordagem global o sistema é analisado como um todo e procura-se um autômato que represente todas as seqüências possíveis de eventos que ele pode gerar e tarefas que pode completar. Para sistemas de maior porte, esta pode ser uma tarefa de grande complexidade. Por outro lado, muitos sistemas de interesse prático apresentam características modulares e/ou distribuídas, de modo que podem ser vistos como constituídos de diversos subsistemas, cada qual gerando certos eventos. O comportamento do sistema como um todo é determinado então pelos eventos produzidos nesses subsistemas.

A abordagem local sugere maior facilidade na obtenção de modelos de sistemas de grande porte. Além disso, permite pressupor que alterações num subsistema ou em alguma restrição somente exigirão uma mudança no modelo específico correspondente.

A aplicabilidade da abordagem local para a modelagem de SEDs por autômatos é garantida pela operação de composição de autômatos, como definida a seguir.

- **Composição Síncrona:**

Considerando os seguintes autômatos:

$G1=(\chi_1,\Sigma_1,f_1,x_{01},\chi_{m1})$ e $G2=(\chi_2,\Sigma_2,f_2,x_{02},\chi_{m2})$

A Composição síncrona entre $G1$ e $G2$ é o seguinte autômato:

$$G1||G2=Ac(\chi_1\times\chi_2,\Sigma_1\cup\Sigma_2,f_{1\|2},(x_{01},x_{02}),\chi_{m1}\times\chi_{m2})$$

Onde se tem que:

$$f_{1||2}((x_1,x_2),e)=\left\{\begin{array}{c}(f_1(x_1,e),f_2(x_2,e)) \quad se \quad e\in\Sigma_1\cap\Sigma_2, \quad e\in\Sigma_1(x_1)\cup\Sigma_2(x_2)\\ (f_1(x_1,e),x_2) \quad se \quad e\in\Sigma_1, e\notin\Sigma_2, \quad e\in\Sigma_1(x_1)\\ (x_1,f_2(x_2,e)) \quad se \quad e\in\Sigma_2, e\notin\Sigma_1, \quad e\in\Sigma_2(x_2)\\ Indeterminada \quad caso \quad contrário\end{array}\right\}$$

## 2.4.Teoria de Controle Supervisório

Diversas abordagens têm sido utilizadas no projeto de controle de sistemas de manufatura, destacando a Teoria de Controle Supervisório (TCS) (RAMADGE e WONHAM, 1989). Esta abordagem permite a síntese automática de controladores a partir da modelagem da dinâmica do sistema em malha aberta (modelos dos

subsistemas sem nenhuma ação de controle) e na modelagem dos requisitos desejados (especificações).

O modelo proposto por Ramadge e Wonham (1989) faz uma distinção clara entre o sistema a ser controlado, denominado planta, e a entidade que o controla, que recebe o nome de supervisor. A planta é um modelo que reflete o comportamento fisicamente possível dos subsistemas. Em geral, este comportamento inclui a capacidade de realizar determinadas atividades que produzam um resultado útil, sem limitar o comportamento desejado. Por exemplo, dois robôs trabalhando em uma célula de manufatura podem ter acesso a um depósito de uso comum, o que pode ser útil para passar peças de um ao outro. No entanto, cria-se com isso a possibilidade física de ocorrer um choque entre ambos, o que é, em geral, indesejável (CURY, 2001).

O papel do supervisor na TCS é, então, o de exercer uma ação de controle restritiva sobre os subsistemas, de modo a confinar seus comportamentos aqueles que correspondem a uma dada especificação. Uma vantagem deste modelo é a de permitir a síntese de supervisores, sendo estes obtidos de forma a restringir o comportamento da planta apenas o necessário para evitar que esta realize ações proibidas. Desta forma, verifica-se uma dada especificação do comportamento podendo ou não ser cumprida e, caso não possa, identificar a parte dessa especificação que pode ser implementada de forma minimamente restritiva. Um critério de aceitação pode então ser utilizado para determinar se o sistema trabalha de maneira satisfatória com a parte implementável da especificação.

### 2.4.1.Controlabilidade

Na TCS existem alguns elementos pertencentes ao $\Sigma$ que são ditos como controláveis $\Sigma c$ e outros ditos como não controláveis $\Sigma nc$, onde $\Sigma = \Sigma c U \Sigma nc$ e $\Sigma_c \cap \Sigma_{nc} = \emptyset$. Em outras palavras, os eventos controláveis são todos aqueles eventos que podem ser inibidos conforme condição de controle imposta pelo projetista e eventos não controláveis são todos aqueles eventos factíveis de ocorrência porém não pode ter nenhuma ação de controle no sentido de inibir sua ocorrência em um determinado instante de tempo.

Considerando $G$ um gerador que representa o comportamento de um sistema físico definido sobre o alfabeto $\Sigma$ . A linguagem $K \subseteq \Sigma^*$ é dita como controlável em relação a $L(G)$ e $\Sigma nc$ , se e somente se após qualquer seqüência de eventos que pertença ao prefixo-fechamento desta linguagem ($s \in \overline{K}$ ) a ocorrência de um evento não-controlável ($\sigma \in \Sigma nc$ ) que seja fisicamente possível ($s\sigma \in L(G)$ ) preserva a seqüência assim formada dentro do prefixo-fechamento desta linguagem ($s\sigma \in \overline{K}$ ). Este conceito pode ser expresso matematicamente como:

$$\overline{K}\,\Sigma nc \cap L(G) \subseteq \overline{K}$$

### 2.4.2.Supervisores

Um supervisor pode ser caracterizado por $S=(E,\Phi)$ , onde $E=(\chi,\Sigma,f,x_0,\chi_m)$ é um autômato onde:

- $\chi$ é um conjunto de estados;
- $\Sigma$ é definido pelo conjuntos dos eventos associado a transição de estados entre os autômatos;
- $f$ é a função de transição dos estados de acordo com $f{:}\chi$ x $\Sigma \rightarrow \chi$ ;
- $x_0$ é o estado inicial, logo $x_0 \in \chi$ ;
- $\chi_m$ é o conjunto de estados marcados, lembrando que $\chi_m \in \chi$ .

De tal forma que reconhece uma linguagem sobre o mesmo conjunto $\Sigma$ de eventos de uma dada planta $G$ *e* $\Phi$ é um mapeamento entre o par (estados de $E$ , eventos de $\Sigma$ ) e o conjunto {desabilitado, habilitado}. O comportamento em malha fechada de um sistema controlado por um supervisor $S=(E,\phi)$ é representado por um autômato $S/G$ , onde o comportamento fechado, verificado pela linguagem $L(S/G)$ , permite a ocorrência de uma cadeia de eventos se esta for aceita por $G$ e $E$ , e se cada elemento da cadeia estiver habilitado por $\phi$ , a supervisão é denotado por $L_m(S/G)=L(S/G)\cap L_m(G)$ .

Pode-se dizer que um dado supervisor $S$ é próprio, ou não bloqueante, se para uma planta definida por $G$ não ocorrer o bloqueio do sistema em malha fechada, ou seja, se $\overline{L_m(S/G)}=L(S/G)$ .

### 2.4.3.Controle Monolítico

A TCS, formulada inicialmente por Ramadge e Wonham (1989), o SED a ser controlado é representado por uma linguagem gerada $L$ e por uma linguagem marcada $L_m$. Assume ainda que a planta $G$ é modelada por um autômato, dessa forma, as linguagens $L(G)$ e $L_m(G)$ podem conter cadeias ou palavras que possam ser indesejáveis, haja visto que pode possuir algum(s) evento(s) que violam alguma condição de controle que se deseja impor ao sistema físico. Pela junção de uma estrutura de controle é possível modificar a linguagem gerada $L(G)$ do sistema dentro de certos limites, evitando assim as cadeias indesejadas.

O autômato $G$ modela então o comportamento do SED sem nenhuma ação de controle. Para a realização do controle monolítico, o projetista de um SED visa encontrar um único controlador a fim de habilitar ou desabilitar um conjunto de eventos pertencentes ao $\Sigma$.

A ação de controle visa à modificação do comportamento do sistema a fim de realizar certas limitações físicas, operacionais, de segurança e/ou lógicas e deve ser entendida como uma restrição do comportamento a um subconjunto de $L(G)$. Para alterar o comportamento introduz-se um supervisor, denominado de $S$.

Dentro desta abordagem, considera-se que o supervisor $S$, definido sobre o mesmo alfabeto $\Sigma$, interage com a planta $G$, onde $S$ observa os eventos ocorridos em $G$ e define que eventos, dentre os fisicamente possíveis de ocorrerem no estado atual, são permitidos de ocorrerem a seguir. Sob este aspecto, a forma de controle é dita permissiva, no sentido que eventos inibidos não podem ocorrer e os autorizados não ocorrem obrigatoriamente.

De acordo com a habilitação ou desabilitação de eventos, o SC consegue moldar a linguagem $L(G)$. O sistema deverá acompanhar a ocorrência dos eventos, onde para tal deverá possuir um único estado ativo do supervisor monolítico a cada instante de tempo e a sua evolução dar-se-á através da ocorrência de eventos de forma seqüencial. Na figura 2.3 visualiza-se o funcionamento de um supervisor monolítico.

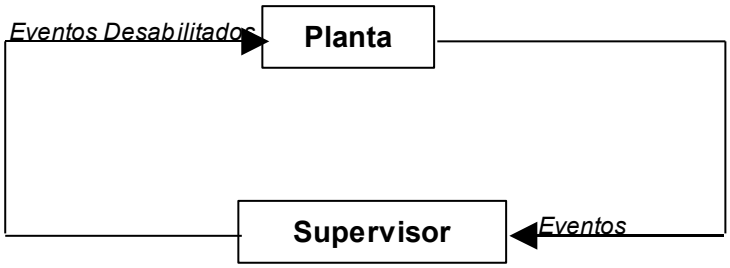


**Figura 2.3** Supervisor Monolítico.

### 2.4.4.Síntese de supervisores monolítico

Para a realização da síntese de um determinado controlador monolítico podem-se usar as noções de controlabilidade e de $L_m(G)$, onde uma linguagem $K \subseteq \Sigma$ é uma linguagem controlável de $L(G)$, quando $\overline{K}\,\Sigma_{nc} \cap L(G) \subseteq \overline{K}$, isso pode ser traduzido como: a ocorrência de um evento controlável e fisicamente passível de ocorrer após uma cadeia de $\overline{K}$, mantém a seqüência no conjunto de $\overline{K}$.

Diz-se que uma linguagem $K$ é $L_m(G)$-fechada se e somente se $K = \overline{K} \cap L_m(G)$, isto é, se todos os seus prefixos que são palavras de $L_m(G)$ também forem palavras de $K$.

Nem sempre é possível a concepção de um supervisor que possa garantir o comportamento do sistema físico, pois pode-se desejar desabilitar eventos não controláveis, para atender uma determinada necessidade de controle, impossibilitando assim tal concepção. Quando um comportamento especificado não possa ser realizado, pode desenvolver um supervisor próprio que possa atender as especificações de forma minimamente restritiva. Neste caso, o controle monolítico objetiva sintetizar um supervisor $S$ para uma linguagem especificada $K \subseteq \Sigma$, tal que $L_m(S/G) = SupC(K,G)$. Em contrapartida se a linguagem formada pelo $SupC(K,G)$ não for aceitável, diz-se que o problema em questão, não possui solução.

### 2.4.5.Modularidade

Considerando $L_{1,} L_2 \subseteq \Sigma$ *, pode-se afirmar que $\overline{L_1 \cap L_2} \subseteq \overline{L_1} \cap \overline{L_2}$, ou seja, o prefixo de uma cadeia comum a $L_1$ e $L_2$ também é um prefixo de $L_1$ e de $L_2$,

diz-se então que $L_1$ e $L_2$ são não conflitantes se $\overline{L_1 \cap L_2} = \overline{L_1} \cap \overline{L_2}$ . Isso quer dizer que duas linguagens são ditas como modulares se e somente se toda a vez que compartilharem um dado prefixo também compartilharem uma palavra contendo esse prefixo.

### 2.4.6.Controle modular local

Quando um grande número de tarefas a ser executadas pelo sistema de controle, a abordagem de Ramadge e Wonham (1989) pode ter um desempenho computacional bastante desfavorável, uma vez que considera a obtenção de um único controlador que observa e atua sobre toda a planta. Uma forma de diminuir a complexidade computacional do problema é dividir a tarefa de controle em várias sub-tarefas ou sub-rotinas que conforme observado na figura 2.4, pode-se explorar a modularidade natural da planta, desta forma, Queiroz e Cury (2002) estende o modelo de Ramadge e Wonham (1989), criando assim a denominada abordagem modular local. Esta abordagem sugere uma arquitetura de controle distribuída em que cada módulo de controle atua somente sobre os subsistemas envolvidos.


**Figura 2.4** Funcionamento dos supervisores.

Na figura 2.5 visualiza-se a arquitetura proposta por Queiroz e Cury (2002b), onde pode-se verificar a existência de alguns níveis para a concepção do SC, representados por:

- Supervisores Modulares: Molda a seqüência de eventos conforme especificações imposta pelo projetista;

- Sistema Produto: Responsável pelo acompanhamento da planta e criação, conforme sinais factíveis de ocorrência, dos eventos a serem enviados para os supervisores, também responsável pela criação dos comandos conforme sinal de habilitação dos supervisores;
- Seqüências Operacionais: Responsável pela seqüência de tarefas a serem executadas.

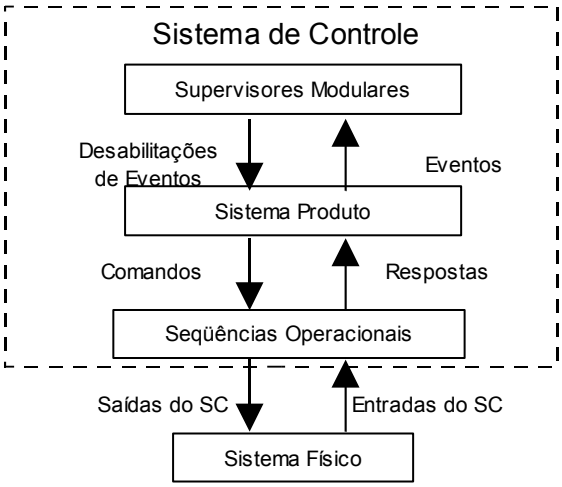


**Figura 2.5** Arquitetura de controle proposta por QUEIROZ e CURY.

Conforme descrito por Vieira (2004), existem alguns aspectos observados no sistema de controle proposto por Queiroz e Cury, dentre os quais enunciam:

- Preserva a modularidade do sistema físico. A inclusão de um subsistema físico não altera a implementação dos módulos do sistema produto e seqüências operacionais relacionadas aos demais subsistemas;
- Preserva a modularidade das especificações. Se a abordagem de síntese dos supervisores é a modular local a inclusão ou exclusão de uma dada especificação não altera a implementação dos supervisores relacionados às demais especificações;
- Com a implementação de um nível correspondente ao Sistema Produto, permite implementar o comportamento livre de cada subsistema físico, individualmente, evitando assim a explosão do número de estados;

- Ao prever que o detalhamento das funções e atividades a serem realizadas por cada subsistema físico seja considerado no nível das seqüências operacionais, permite que sejam utilizados modelos de alto grau de abstração para descrever o comportamento livre destes subsistemas, evitando assim a explosão de estados durante a fase de síntese dos supervisores.

## 2.5.Técnicas de Implementação do Sistema de Controle

Neste tópico serão abordadas algumas considerações a respeito da implementação de um SC, considerando para tal o embasamento matemático levantados nas seções anteriores deste capítulo, para isso será adotado a seguinte consideração verificada na figura 2.6, onde se deseja controlar o início de operação de cada uma das máquinas representada aqui neste por Maq1, Maq2 e Maq3, considerando que o Buffer de tal sistema é unitário, sendo assim deve-se controlar o *overflow* e *underflow*.

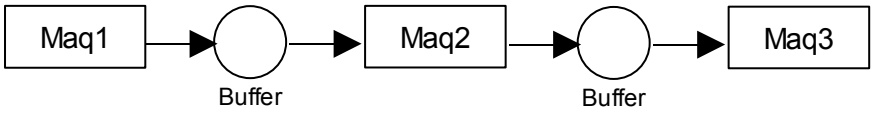


**Figura 2.6** Sistema físico a ser controlado.

Considera-se como eventos controláveis o conjunto formado pelo alfabeto $\Sigma_c = \{\alpha_1, \alpha_2, \alpha_3\}$ e os eventos não controláveis representado pelo conjunto $\Sigma_{nc} = \{\beta_1, \beta_2, \beta_3\}$, onde $\alpha_i$, para $i = 1,2,3$, significa o inicio de cada atividade das máquinas e $\beta_i$, para $i = 1,2,3$, traduz-se como o término da atividade para cada uma das máquinas. Para tal sistema tem-se a representação do grafo verificado na figura 2.7.

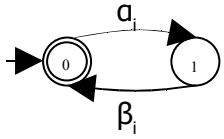


**Figura 2.7** Autômato do sistema físico.

Para atender as condições de controle impostas ao sistemas, deve-se criar as seguintes especificações, descritas e verificadas na figura 2.8, especificações essas para atender o requisito de controle a respeito de *overflow* e *underflow*.

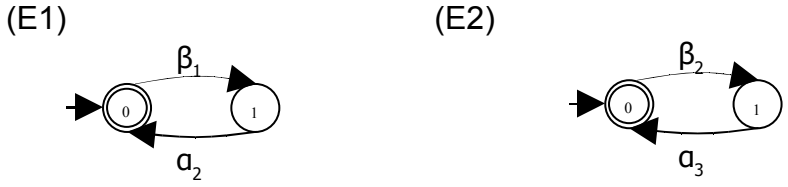


**Figura 2.8** Especificações.

Para a aplicação inicial dois supervisores são sintetizados, cada um com uma finalidade específica, a partir das especificações selecionadas e apresentadas na figura 2.8. Por exemplo, considerando a especificação $E_1$, a planta local é obtida através da composição síncrona dos autômatos correspondentes aos subsistemas que compartilham eventos com esta especificação ($G_{loc,1} = G_1 \,||\, G_2$). A especificação local é obtida através da composição da especificação genérica $E_1$ com a correspondente planta local ($E_{loc,1} = E_1 \,||\, G_{loc,1}$). Pode-se então calcular a máxima linguagem controlável contida na especificação, que é SupC ($E_{loc,1}$, $G_{loc,1}$). Em seguida, através de um algoritmo de minimização de supervisores (VAZ & WONHAM, 1986), obtém-se um supervisor com um menor número de estados e com a mesma ação de controle. A tabela 2.1 apresenta os supervisores calculados a partir dos modelos dos subsistemas e das especificações selecionados para a aplicação inicial.

| Especificação genérica | Planta Local | Especificação local | Máxima linguagem controlável (Supervisor Modular) |
|---|---|---|---|
| $E_1$ | $G_{loc,1} = G1 \,||\, G2$ | $E_{loc,1} = G_{loc,1} \,||\, E_1$ | SupC($G_{loc,1}$ , $E_{loc,1}$ ) |
| $E_2$ | $G_{loc,2} = G2 \,||\, G3$ | $E_{loc,2} = G_{loc,2} \,||\, E_2$ | SupC($G_{loc,2}$ , $E_{loc,2}$ ) |

Tabela 4.1 Modelos utilizados no procedimento de síntese e supervisores locais resultantes.

Por exemplo, a figura 2.9 apresenta o supervisor SupC($G_{loc,1}$ , $E_{loc,1}$), resultante da especificação $E_1$. A linha tracejada vermelha indica a ação de controle do supervisor, que é desabilitar eventos controláveis dos subsistemas G1 e G2.

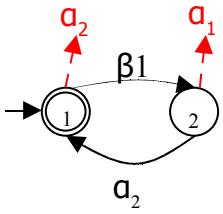


**Figura 2.9** Supervisor SupC($G_{loc,1}$ , $E_{loc,1}$) para a aplicação inicial.

Com o que já foi exposto para a concepção do SC, deve seguir os seguintes passos: primeiramente constroem os supervisores, de acordo com a figura 2.10, que trata de como irá ocorrer a evolução dos estados dos autômatos, de acordo com a ocorrência dos eventos, em seguida implementa-se os sinais de desabilitação, conforme demonstrado na figura 2.11, sinais esses de suma importância para o controle, haja visto que tal sinal corresponde a qual tarefa não será executada em um determinado instante. Concluído esta etapa, deve-se gerar os sinais de desabilitação de módulos, correspondente à figura 2.12, sinais esses vitais para a garantia de que só deverá ocorrer um único evento em um determinado instante de tempo, conforme mencionado e discutido na seção 2.4.

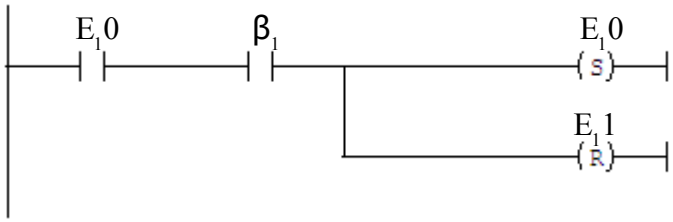


**Figura 2.10** Implementação dos supervisores.


**Figura 2.11** Estrutura para sinais de desabilitação.


**Figura 2.12** Estrutura para sinais de desabilitação de evolução de módulos.

Verificada a implementação do relato anterior deve-se gerar os sinais $\alpha_i$ e $cmd\alpha_i$, sendo que esse último sinal pode ser mantido em mais de um passo para a realização de sincronismo com outros dispositivos, porém os sinais internos $\alpha_i$ devem ser ativos em apenas um único ciclo, para a geração de tais sinais, bem como o sinal $G_i evt$, pode ser verificado no decorrer deste com a figura 2.13.

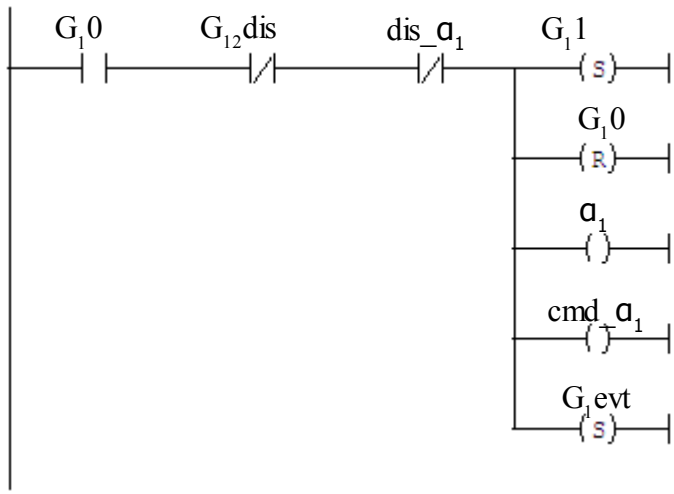


**Figura 2.13** Estrutura geração dos sinais de habilitação, controle do módulo do sistema produto e controle dos estados.

Para gerar os sinais $\beta_i$ e $rpl\beta_i$, verifica-se na figura 2.14 da presente geração, que trata justamente desta questão dos sinais oriundos do sistema físico e levando em consideração, novamente falando, o que foi discutido na seção 3.2 onde refere-se a impossibilidade de possuir dois sinais simultâneos em um mesmo instante de tempo.

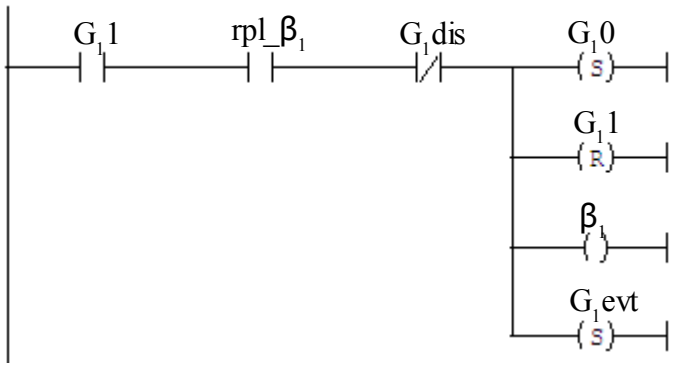


**Figura 2.14** Estrutura geração dos sinais de retorno, controle do módulo do sistema produto e controle dos estados.

## 2.6.Resumo do capítulo

Nesse capítulo verificaram-se aspectos relacionados a algumas formas de modelagem, destacando as linguagens formais e autômatos, onde foi relatada a fundamentação matemática para tal aplicação, bem como algumas características de SEDs e a TCS proposta por Ramadge e Wonham além da extensão de tal modelo por Queiroz e Cury.

Discutiu-se alguns aspectos apontados por Vieria (2004) referente à implementação e algumas limitação do modelo original proposto por Ramadge e Wonham, alguns aspectos relacionados na adoção do modelo proposto por Queiroz e Cury.

# 3
# Abordagem de Desenvolvimento

Esta seção apresenta uma abordagem de desenvolvimento de sistemas de controle para processos reconfiguráveis no contexto da manufatura ágil. A figura 2 ilustra o ciclo de desenvolvimento, caracterizado em três etapas: modelagem, síntese e implementação. Na etapa de modelagem seleciona-se, a partir de bibliotecas de subsistemas e de especificações, um conjunto de modelos para a representação do sistema real e da aplicação, respectivamente. Na etapa de síntese, os modelos são utilizados na geração dos supervisores modulares, de acordo com a TCS (RAMADGE e WONHAM, 1989) e a Teoria de Controle Modular (QUEIROZ e CURY, 2000). Na etapa de implementação, os três níveis da estrutura de controle (supervisores modulares, sistema produto e as seqüências operacionais) são integrados e implementados gradativamente em três fases: simulação; inserção das tecnologias de controle e comunicação; acoplamento gradativo dos subsistemas.

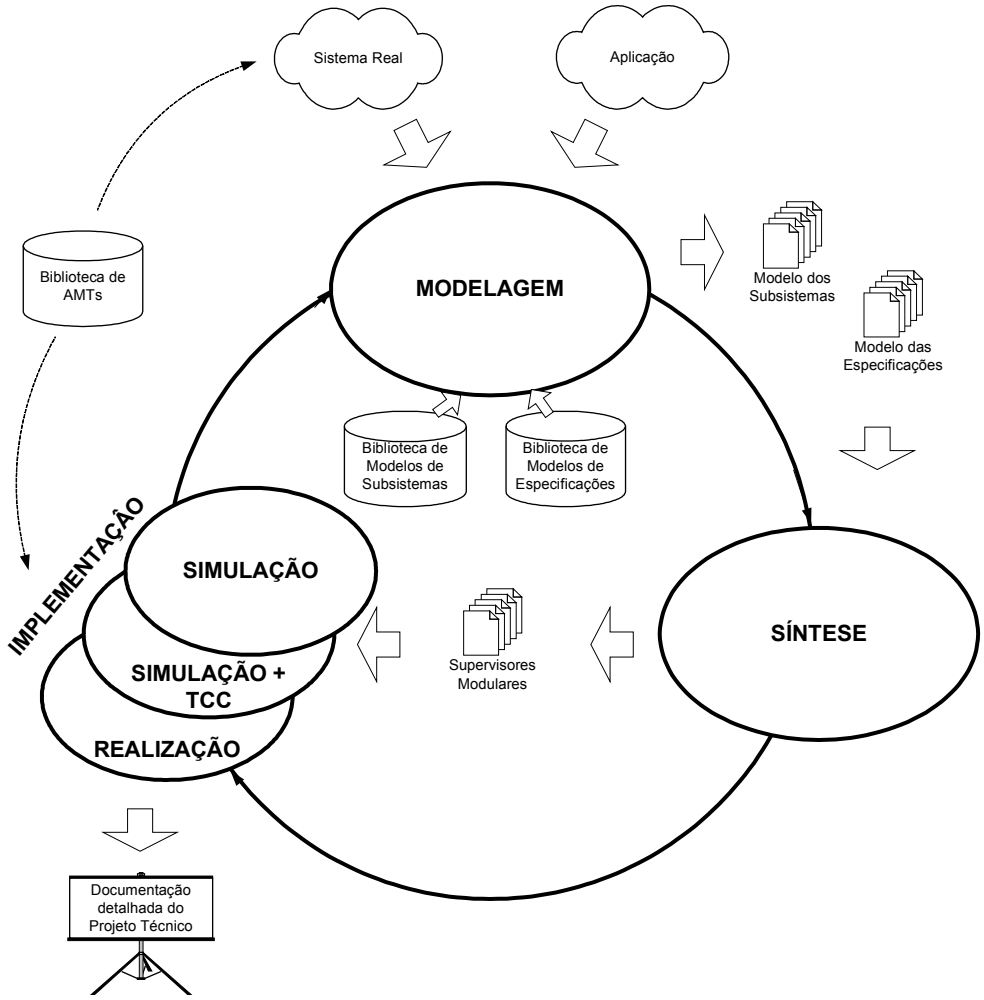


**Figura 3.1** Ciclo de desenvolvimento do sistema de controle para processos reconfiguráveis.

O desenvolvimento do sistema de controle ocorre ciclicamente em três etapas de modelagem, síntese e implementação até o atendimento da aplicação demandada para o sistema real, resultando no sistema automatizado e integrado. Esta forma de desenvolvimento permite uma revisão contínua dos resultados obtidos em cada etapa. Assim, o projetista poderá receber uma nova aplicação (por exemplo, uma necessidade da reconfiguração dos processos) e selecionar novos modelos de especificações ou de subsistemas que atendam adequadamente esta nova aplicação.

Uma biblioteca de Tecnologias Avançadas de Manufatura – AMTs da suporte a base tecnológica na definição das seqüências operacionais relacionadas aos subsistemas, da implementação dos supervisores modulares e das necessidades de

reconfiguração dos processos. Segundo Gouvêa da Costa et al. (2000), o conceito de AMT abrange aparatos de base numérica e computacional (software e hardware), projetados para realizar ou suportar atividades e tarefas da manufatura. As tecnologias de comunicação, controladores programáveis, sensores e atuadores industriais, dentre outros, são exemplos de AMTs.

## 3.1.Etapa de Modelagem

De acordo com a abordagem modular, o funcionamento livre (sem controle) dos diversos subsistemas pode ser modelado como um conjunto de autômatos assíncronos (sem eventos em comum). Dessa forma, Queiroz e Cury (2002) afirmam ser viável a obtenção de uma representação por Sistema Produto, conforme discutido na seção anterior.

i. Identificar o conjunto de subsistemas envolvidos no sistema de manufatura;
ii. Construir o autômato *Gi* de cada subsistema *i* envolvido de forma mais sintética possível;
iii. Calcular a mais refinada Representação por Sistema Produto (RSP), fazendo-se a composição dos subsistemas síncronos;
iv. Modelar cada especificação isoladamente, considerando apenas os eventos relevantes;

De acordo com o ciclo de desenvolvimento apresentado na figura 3.1, inicialmente deve-se selecionar modelos adequados relacionados a todos os subsistemas que compõem o sistema real. Em seguida, selecionar modelos que representem as especificações a serem aplicadas ao sistema, sendo que estas definem a forma de coordenação de todos os subsistemas. Entretanto, a tarefa de construir modelos apropriados que representem os subsistemas, bem como cada uma das especificações, não é tarefa simples e requer certa experiência na modelagem de sistemas de manufatura (CASSANDRAS e LAFORTUNE, 1999). Assim, apesar da vantagem da TCS residir na síntese automática de supervisores, a construção de modelos para o sistema real e especificações poderá depender da

experiência e inspiração do projetista, comprometendo a confiabilidade, o tempo necessário e o custo global do desenvolvimento.

Em muitos casos a modelagem dos subsistemas consiste em uma tarefa relativamente simples, uma vez que a própria configuração espacial do sistema real (por exemplo, sistema de manufatura composto por células e estações) permite que o projetista selecione os diversos subsistemas existentes. A atenção aqui deve ser considerada na identificação dos estados de cada um dos subsistemas. Deve-se compatibilizar as funções de coordenação esperadas pelo sistema de supervisão com a correta identificação dos estados dos diversos subsistemas. De maneira geral, é possível identificar os seguintes estados dos subsistemas:

- Estado inativo (geralmente estado inicial);
- Estados ativos (um subsistema eventualmente pode ter diferentes estados de funcionamento);
- Estado de quebra ou falha.

De acordo com a aplicação demandada ao sistema de manufatura, constroem-se especificações que a atendam. Em seguida, procura-se construir um supervisor local para cada especificação, modelando-o apenas em termos dos subsistemas afetados por sua ação. De posse dos supervisores obtidos no processo de síntese, garante-se que a aplicação demandada será atendida.

Na modelagem das especificações, a construção do conjunto de modelos corresponde aos requisitos a serem impostos aos subsistemas. Cassandras e Lafortune (1999) definem classes de especificações comumente presentes em SED. Os autores consideram quatro casos em que se baseiam os modelos de especificações: estados proibidos, alternância de eventos, cadeias ilegais e refinamento de estados. No primeiro caso, identifica-se no modelo do sistema real quais estados não são factíveis de ocorrer, em função de restrições físicas ou de segurança. O modelo da especificação é simplesmente obtido pela exclusão destes estados do sistema. No segundo caso, o requisito de coordenação impõe a alternância entre eventos. Por exemplo, a necessidade de alternar dois eventos *a* e *b*, com *a* ocorrendo primeiro, leva a construção de um autômato de dois estados que captura esta alternância. No terceiro caso, identifica-se como ilegais todas as

cadeias do modelo do sistema real que contêm determinadas sub-cadeias. No último caso, necessita-se memorizar como um determinado estado do sistema foi alcançado de forma a especificar qual comportamento futuro é admissível. Deve-se então refinar tal estado em quantos estados forem necessários.

Em trabalhos anteriores, Santos (2004) propõem a criação de bibliotecas de modelos de especificações relacionadas à configuração física de sistemas de manufatura. O principal aspecto explorado é o tipo de transporte utilizado entre estações de trabalho. Dessa forma, a utilização de transportadores síncronos e assíncronos (GROOVER, 2001) conduz a um grupo particular de especificações. A utilização de posições intermediárias (em função de restrições físicas ou previsão de atraso de transferência de produtos) entre estações de trabalho também gera modelos de especificações particulares. Santos et al. (2004) explora outras diversas configurações possíveis em sistemas de manufatura e a cada uma destas um modelo de especificação é associado.

Além da utilização de bibliotecas, a etapa de modelagem pressupõe que todos os subsistemas que compõem o sistema de manufatura estão identificados e tem, cada um destes, uma seqüência operacional associada.

### 3.2.Etapa de Síntese

v. Obter a planta local para cada especificação compondo-se os subsistemas da RSP que tenham eventos em comum com ela;

vi. Calcular a linguagem de cada planta local que satisfaça a especificação, através do produto síncrono da cada planta local com sua respectiva especificação;

vii. Calcular a máxima linguagem controlável contida em cada especificação local;

viii. Verificar a modularidade local das linguagens resultantes;

ix. Se não forem modulares, procurar resolver o problema de não modularidade por outra abordagem;

x. Se forem modulares, implementar um supervisor local para cada linguagem controlável.

Esta etapa consiste em aplicar o procedimento de síntese proposto por Ramadge e Wonham (1989) e Queiroz e Cury (2000) (2000b). A partir dos modelos selecionados na etapa anterior, supervisores modulares são sintetizados. Considerando-se que a etapa posterior tem como objetivo implementar os supervisores em plataformas industriais (por exemplo, PLC), é necessário ainda aplicar algoritmos de redução de supervisores (VAZ e WONHAM, 1986) de forma a obter-se supervisores reduzidos (menor número de estados) com a mesma ação de controle. Isso permite uma menor quantidade de memória e uma melhor legibilidade do programa de controle.

Identificam-se ainda as ferramentas que tratam da síntese da estrutura de controle, baseadas em formalismos matemáticos específicos. Como exemplo, o processo de síntese do sistema de controle na TCS, pode ser realizado através do TCT (WONHAM, 1999), GRAIL (RAYMOND e WOOD, 1996), DESCO (FABIAN e HELLGREN, 2000), UKDES (CHANDRA et al., 2002), e VER (BALEMI et al., 1993). O GRAIL, originalmente concebido como uma ferramenta de manipulação de autômatos (RAYMOND e WOOD, 1996), tem agora algoritmos desenvolvidos pelo grupo de Automação da UFSC (CURY, 2001) que tratam todo o processo de síntese de controladores baseado na abordagem de Ramadge e Wonham (1989). TCT

A ferramenta computacional SUPREMICA (AKESSON, 2002) possibilita ainda a tradução de uma linguagem formal (autômatos) numa linguagem normalizada de programação de controladores programáveis. Assim, é possível, a partir de uma estrutura em autômatos, gerar o código de controlador de acordo com a IEC 61131-3 (1998).

### 3.3.Etapa de Implementação

Esta etapa consiste de três fases em que a estrutura de controle é progressivamente acoplada ao sistema real. Para a realização destas três fases é utilizada uma plataforma computacional de suporte a geração dos códigos dos supervisores a partir dos arquivos do TCT obtidos na etapa de síntese.

A primeira fase consiste na simulação integrada dos subsistemas reais em aplicativos específicos (ex, Arena, EM-Plant) e dos supervisores e sistema produto na plataforma Code Generator.Faz-se a simulação dos três níveis (supervisores

modulares, sistema produto e seqüências operacionais) da estrutura de controle. A integração do aplicativo de simulação com a plataforma Code Generator é realizada através da comunicação entre os subsistemas e o sistema produto envolvendo os sinais de comando e resposta, respectivamente. No simulador estão implementados os subsistemas de acordo com a configuração do sistema real e que foi utilizada na geração do sistema produto. Desta forma, a simulação ocorre de acordo com as restrições impostas pelos supervisores modulares que estão implementados na plataforma Code Generator. Por exemplo, o simulador somente iniciará a simulação de um subsistema i quando o respectivo comando for enviado pela plataforma Code Generator. O resultado da simulação permite avaliar a completude ou possíveis erros dos modelos dos subsistemas e das especificações, além dos resultados quantitativos da simulação. O projetista pode iniciar o funcionamento do nível dos supervisores modulares e sistema produto e acompanhar a evolução de estados e ações de controle associadas.

Numa segunda fase, tecnologias de controle e comunicação são inseridas ao ambiente de simulação com a finalidade de testar e validar a topologia física de controle distribuído das seqüências operacionais (nível inferior da estrutura de controle). As tecnologias de controle simulam o conjunto de seqüências operacionais, sendo que o projetista pode associar a estas, rotinas de temporização e de interface com botoeiras e sinalizadores. Por exemplo, conhecendo-se o tempo de execução de atividade de um determinado subsistema, pode-se simular a seqüência operacional através de um código de temporização.

Na terceira fase, os dispositivos que implementam as seqüências operacionais são progressivamente acoplados aos respectivos sensores e atuadores dos subsistemas reais. O projetista pode então substituir gradativamente um subconjunto de SO simuladas por SO reais até a implementação completa da estrutura de controle. Nesta fase também ocorre a tradução do conjunto de supervisores modulares e o sistema produto (conjunto de modelos dos subsistemas) em linguagem própria de programação.

# 4
# Modelagem da Ferramenta Case Code Generator

A modelagem a seguir trata da concepção de uma ferramenta de auxílio denominada de Code Generator, tal ferramenta auxilia no uso da abordagem de desenvolvimento comentada no capítulo anterior, que trata dos seguintes aspectos no desenvolvimento de SC:

i. Em uma primeira etapa o projetista deverá realizar a modelagem do sistema a ser controlado;

ii. Finalizada tal tarefa deve-se iniciar a etapa de síntese, essas duas etapas podem ser realizadas através de algumas das ferramenta de síntese citadas anteriormente, e inseridos no sistema Code Generator;

iii. Para o processo de implementação do SC a ferramenta Code Generator auxilia o projetista com as etapas de simulação e implementação física.

## 4.1.Relacionamento entre os Elementos do Método Proposto

A ferramenta Code Generator possui os seguintes módulos e sua integração ao longo do desenvolvimentos de novos projetos de sistemas lógicos aplicados a indústria, conforme verificado na figura 4.1 pode-se visualizar 3 (três) grandes módulos:

- Ambiente de Desenvolvimento;
- Ambiente de Validação;
- Ambiente de Auxílio à Implementação.

Esses módulos estão de acordo com o que foi relatado e sugerido por Moore et al. (2003), onde tais módulos deverão possuir uma base de dados integrada para possuir uma maior eficiência, ou seja, as informações inseridas em um dos módulos e armazenadas pela ferramenta será utilizada pelos demais módulos.

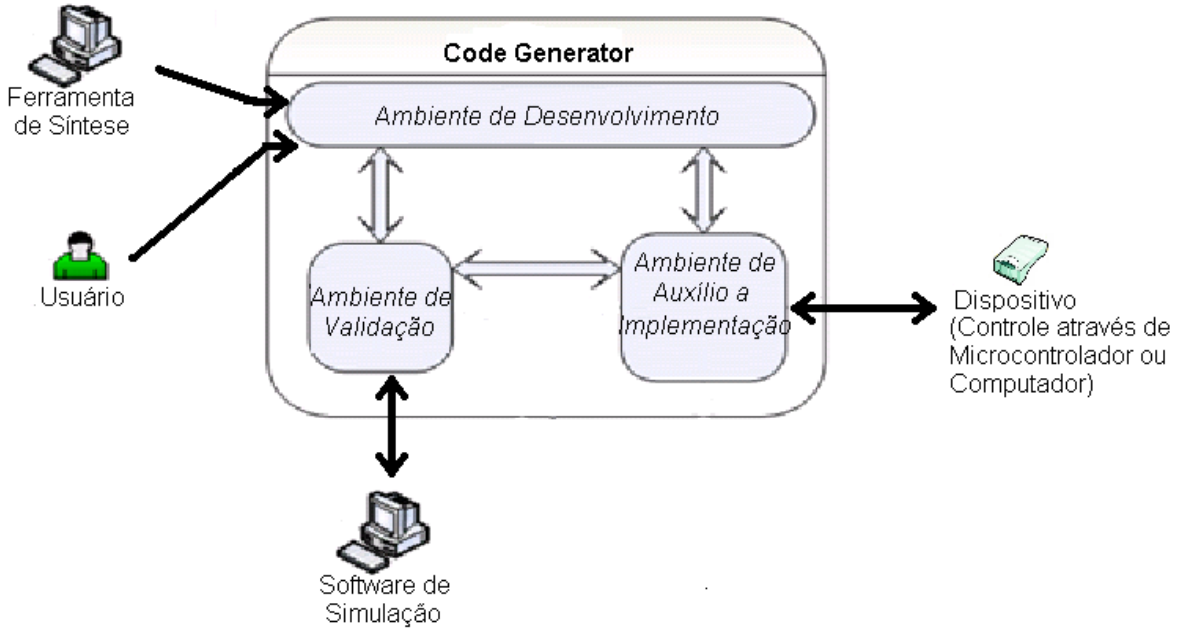


**Figura 4.1** Diagrama de blocos dos módulos do sistema Code Generator.

### 4.1.1.Ambiente de Desenvolvimento

Consiste basicamente do local onde são inseridos os modelos formais dos subsistemas e das especificações do sistema de manufatura, a partir da leitura das tabelas do GRAIL, modelos estes baseados na teoria de linguagens formais e autômatos. De acordo com a abordagem utilizada, discutida no capítulo 3, a aplicação desta pressupõe a construção de modelos adequados que representam a planta (formada geralmente por diversos subsistemas) e as especificações que se deseja impor a ela. Após a construção destes modelos, pode-se sintetizar controladores de forma automática que garantam o correto funcionamento definido pelo conjunto de especificações. As ferramentas computacionais discutidas na seção 1.2 na sua maioria tratam especificamente do processo de síntese, que engloba a modelagem da planta e especificações em autômatos. Entretanto, são

plataformas dedicadas a estas operações, não fornecendo ao projetista o suporte necessário para lidar com a complexidade inerente a esta etapa do ciclo de desenvolvimento.

### 4.1.2.Ambiente de Validação

Consiste em um módulo de validação do projeto através de um ambiente de simulação, conforme proposto no capítulo anterior, para desenvolvimento de sistemas lógicos aplicados na indústria sugere o uso de uma ferramenta de simulação para a validação do SC gerado, sendo possível alterações e re-projeto dos modelos dos subsistemas e das especificações encontradas para atender as necessidades do sistema físico. Tal módulo é capaz de reproduzir ações e representá-las em um ambiente de validação através de simulações dos controladores gerados anteriormente. Para alcançar esta funcionalidade, o projetista deverá utilizar-se de software tal como Em-Plant ou Arena.

### 4.1.3.Ambiente de Auxilio a Implementação

O módulo dá suporte as últimas atividades do desenvolvimento do sistema automatizado de manufatura. Estas atividades são relacionadas a implementação, validação e testes do sistema de controle obtido a partir dos módulos descritos anteriormente. Dessa forma, é através desse módulo que é possível a realização física do projeto em questão, quer seja através do controle pelo microcontrolador quer seja através do computador.

## 4.2.Diagrama de Entidade e Relacionamento

Diagrama de Entidade e Relacionamento (DER) é uma etapa do processo de modelagem do software de suma importância na modelagem de um software envolvendo um sistema gerador de banco de dados, pois de acordo com a modelagem proposta pode-se ter uma maior eficiência computacional, uma vez que a tarefa de comunicação com a base de dados é uma das tarefa que consome um recurso considerável no uso do processador.

Na figura a seguir, pode observar as tabelas envolvidas para a concepção do software Code Generator, bem como os relacionamentos entre as mesmas, essas tabelas serão implementadas em um sistema gerador de banco de dados, para uma melhor compreensão considera-se por exemplo a tabela *TB100_EVENT*, tal tabela possui um relacionamento, ou seja, troca informações com as tabelas *TB200_PRODUCT_SYSTEM* e *TB300_SUPERVISORS*, sendo a primeiro um relacionamento dito 1 para 1 e a segunda 1 para n, conforme verificado pelas setas que ligam as tabelas, ou seja a do primeiro caso posui apenas uma reta nas duas pontas e no segundo caso possui ramificações na tabela *TB300_SUPERVISORS* indicando que este pode possuir n eventos oriundos da tabela *TB100_EVENT*.

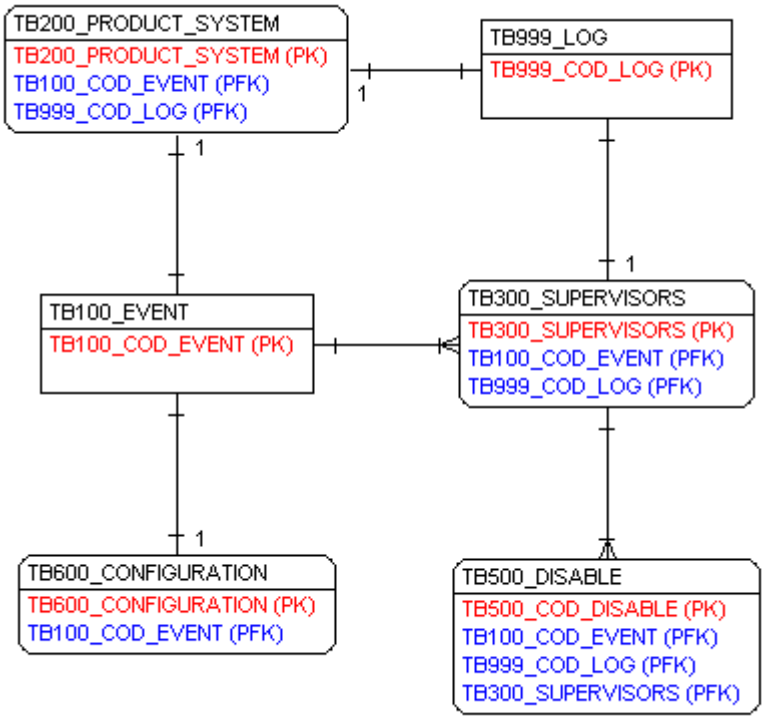


**Figura 4.2** Diagrama de Entidade e Relacionamento.

### 4.3.Diagrama de Contexto

Neste tópico serão apresentados os estímulos do software Code Generator. Na figura 4.3 apresenta os estímulos do software, onde é possível a vizualização de como e quais informações são necessárias para operar o sistema, além de informar quais dados o software deverá retornar para o Microcontrolador, Dispositivo e

software de simulação, em uma forma de exemplificação da funcionalidade de tal diagrama, visualiza-se a entidade externa *Ferramenta de Síntese* e *Software de Simulação,* onde caberá a primeira entidade o fornecimento de todo o modelo da planta bem como os supervisores a fim de codificar o SC pelo Code Generator, concluída esta tarefa o sistema fornecerá a segunda entidade os comandos para execução de determinadas sub-plantas e a simulação fornecerá as respostas obtidas, conforme modelo implementado.

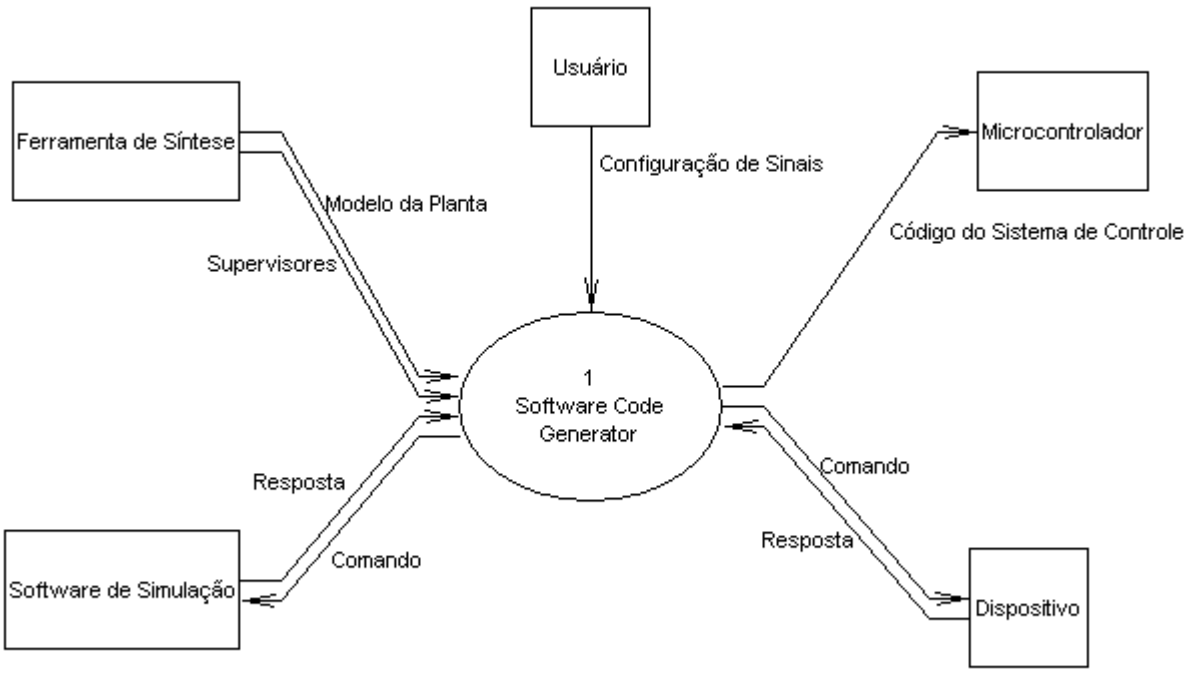


**Figura 4.3** Diagrama de Contexto.

### 4.3.1.Lista de Eventos

Com base no que foi exposto a respeito do diagrama de contexto é possível montar a tabela 4.1 a respeito da lista de eventos, tal tabela é de grande importância para a modelagem de todos os processos do sistema, devendo ainda fazer a seguinte consideração com relação a coluna Tipo: (F) trata-se de um fluxo de dados; (T) Para um evento temporal.

| *Nº* | *Nome do Evento* | *Estímulo* | *Ação* | *Resposta* | *Tipo* |
|---|---|---|---|---|---|
| 1 | Ferramenta de Síntese Cadastra Modelo da Panta | Modelo da Planta | Cadastrar Modelo da Planta | Cadastrado o Modelo da Planta | F |
| 2 | Ferramenta de Síntese Cadastra Supervisores | Supervisores | Cadastrar Supervisores | Cadastrado os Supervisores | F |
| 3 | Usuário Configura Sinais para Controle via Computador | Configuração de Sinais | Cadastrar Configuração de Sinais para Controle via Computador | Cadastrado as Configurações de Sinais | F |
| 4 | É Hora de Comunicar com Software de Simulação | | Comando | Resposta | T |
| 5 | É Hora de Comunicar com Dispositivos | | Comando | Resposta | T |
| 6 | É Hora de Comunicar com Dispositivo (Micro-controlador) | | Enviar Código para o Micro-controlador | Código do Sistema de Controle | T |

**Table 4.1:** Lista de Eventos.

### 4.3.2.DFD de resposta ao evento 1

Neste módulo o sistema irá identificar os diversos eventos que são considerados como importantes na modelagem através dos módulos obtidosdas plantas $Gi$ , estes dados são informados ao sistema através dos arquivos reconhecidos pelo GRAIL, conforme exposto na figura 4.6 que reconhece os eventos a1 e b1, mas além desse reconhecimento o sistema identifica também as mudanças de estados e a evolução do sistema conforme a ocorrência desses eventos.

As informações reconhecidas pelos arquivos do GRAIL são armazenadas no sistema gerador de banco de dados, conforme tabelas visualizadas na figura 4.5, estas informações serão necessárias para a execução dos demais módulos do sistema Code Generator, destacando aqui os modulos de realização física e virtual.

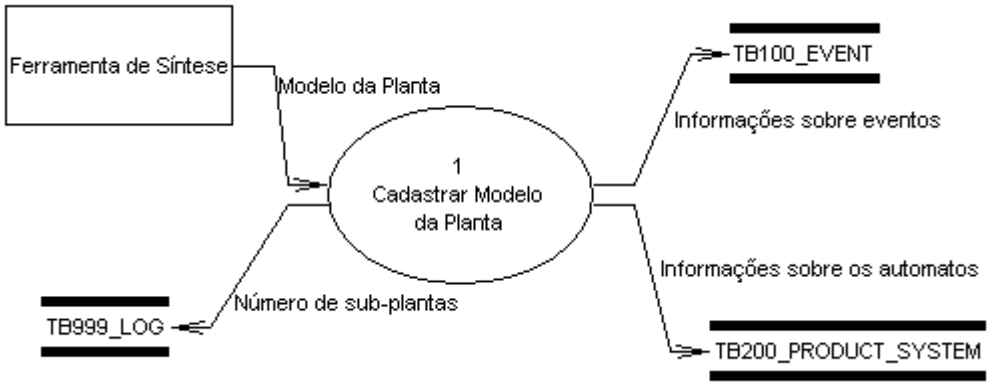


**Figura 4.5** Estrutura da tela de Cadastro do Modelo da Planta.

```
(START) |- 0
0 a1 1
1 b1 0
0 -| (FINAL)
```

**Figura 4.6** Código do autômato.

### 4.3.3.DFD de resposta ao evento 2

Módulo responsável pela identificação dos diversos eventos reconhecidos pelos modelos dos supervisores $S_n$, estes dados são informados ao sistema através dos arquivos do reconhecidos pelo GRAIL, conforme exposto na figura 4.8 que reconhece os eventos b1 e a2, mas além desse reconhecimento o sistema identifica também as mudanças de estados e a evolução do sistema conforme a ocorrência desses eventos, e quais eventos são desabilitados ou inibidos de ocorrência em um determinado estado do supervisor $S$, como por exemplo no estado 0 ocorre a desabilitação do evento a2.

As informações reconhecidas pelos arquivos do GRAIL são armazenadas no sistema gerador de banco de dados, conforme tabelas visualizada na figura 4.7, estas informações serão necessárias para a execução dos demais módulos do sistema Code Generator, destacando aqui os modulos de realização física e virtual.

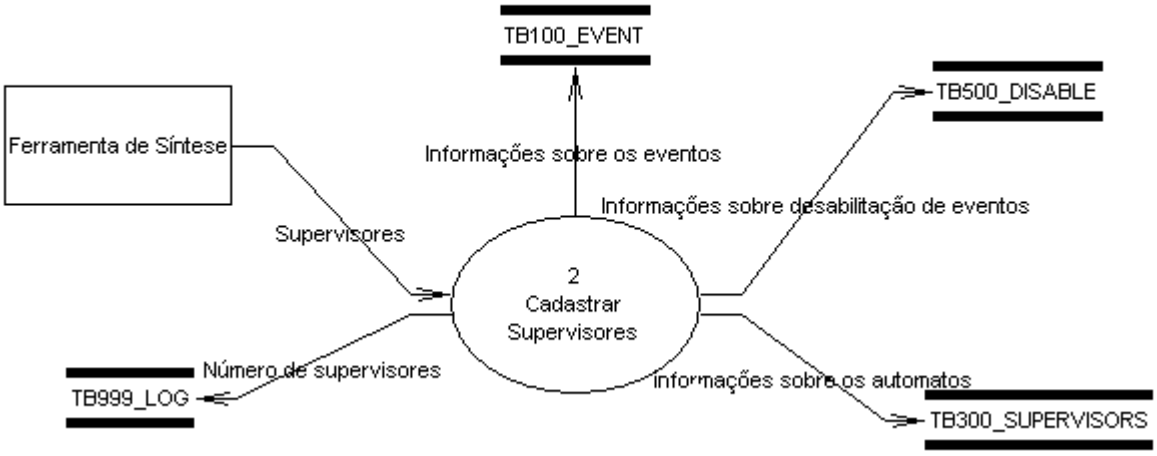


**Figura 4.7.** Estrutura da tela de Cadastro de Supervisores.

```
(START) |- 0
0 a1 1
1 b1 0
0 -| (FINAL)
(SUP)
0 a2 0
1 a1 1
-| (FINAL)
```

**Figura 4.8** Código do autômato.

### 4.3.4.DFD de resposta ao evento 3

Este módulo é responsável pela configuração dos sinais a serem emitidos pelo SC, caso a implementação escolhida seja via computador. É através deste que é possível o envio de sinais referentes aos comandos para as seqüências operacionais implementada em um outro dispositivo industrial, além do recebimento de sinais referentes as respostas e encaminhá-las ao Sistema Produto. Tooda essa comunicação está disponível apenas para comunicação via porta paralela até o momento.

Na figura a seguir, visualiza-se o fluxo de dados e quais as tabelas necessárias bem como informações tramitadas com a execução do referido módulo.

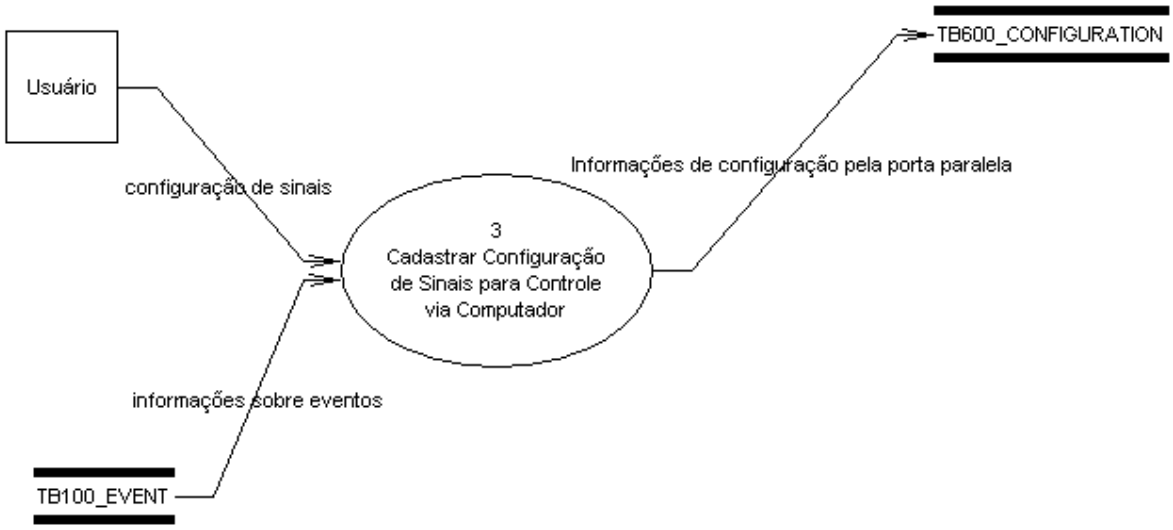


**Figura 4.9** Estrutura da tela de Configuração de sinais pela porta paralela.

### 4.3.5.DFD de resposta ao evento 4

Para a realização deste módulo, a comunicação com o software de simulação será através da rede ethernet, onde o software Code Generator aguarda um sinal para estabelecer uma nova conexão realizado isso o SC começa a enviar sinais ao sistema virtual e receber do mesmo as respostas. A validação do SC desenvolvido será através da identificação da palavra gerada pelo sistema, ou pela própria validação visual em um software de simulação como no caso do Em-Plant ou Arena.

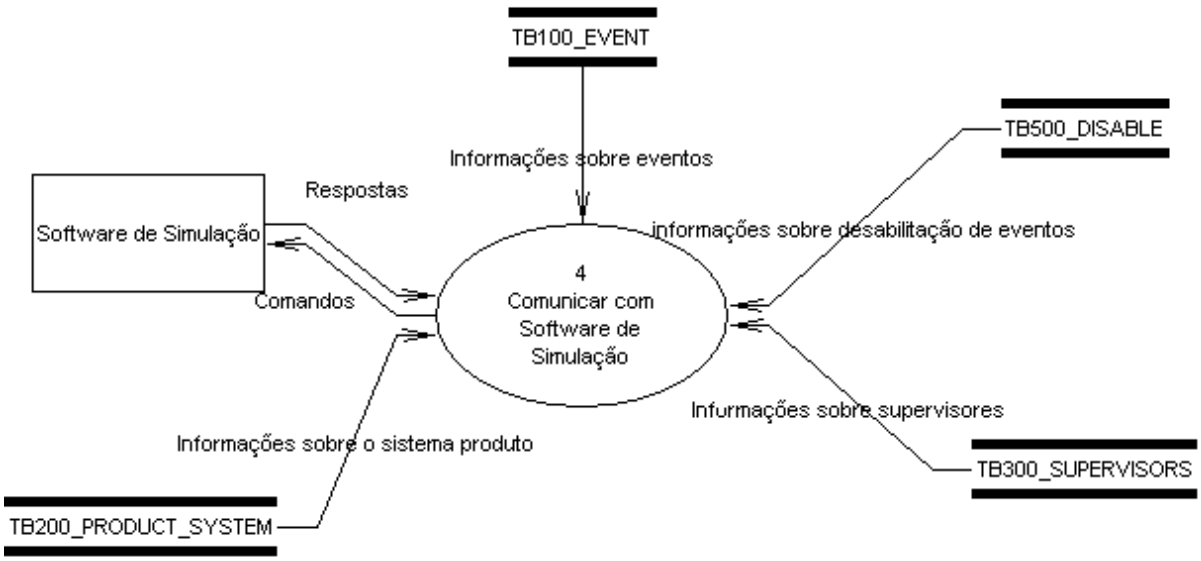


**Figura 4.10** Estrutura da tela para Comunicar com Software de Simulação.

### 4.3.6.DFD de resposta ao evento 5

Módulo responsável pela realização física através do computador, assim como no módulo relatado anteriormente referente a realização virtual, esse módulo captura as informações através de uma base de dados integrada. O SC envia sinais através da porta paralela do computador aos dispositivos industriais conforme modelo vislumbrado pelo projetista.

Tendo como base a comunicação paralela tal módulo fica limitado a 8 (oito) sinais de saídas, ou ainda 8 (oito) sinais controláveis e 5 (cinco) sinais ditos como não controláveis. Na figura a seguir visualiza-se quais tabelas e quais informações são trafegados pelo sistema Code Generator para tal módulo.

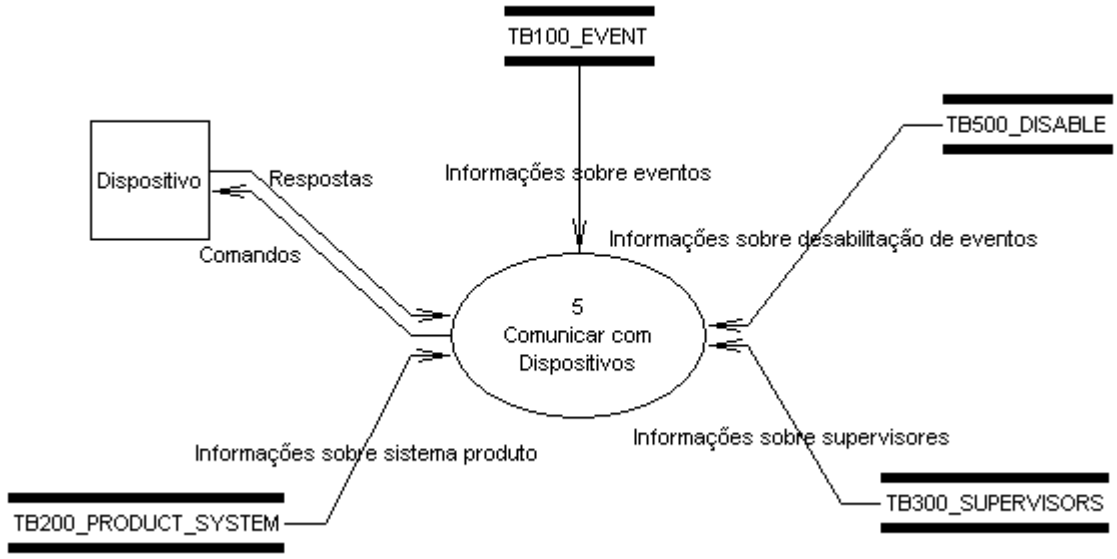


**Figura 4.11** Estrutura da tela para Comunicar com Dispositivos.

### 4.3.7.DFD de resposta ao evento 6

Módulo responsável pela realização física através do micro-controlador, assim como no módulo relatado anteriormente referente a realização virtual, esse módulo captura as informações através de uma base de dados integrada. O software em questão gera um código em assembler para ser implementado em um micro-controlador do tipo PIC da Microchip. A realização do mesmo dar-se-á de acordo como relatado por Costa et all (2005) que preve tal realização através da implementação de apenas 2 níveis da arquitetura proposta por Queiroz e Cury (2002b).

Na figura a seguir visualiza-se quais tabelas e quais informações são trafegados pelo sistema Code Generator para tal módulo.

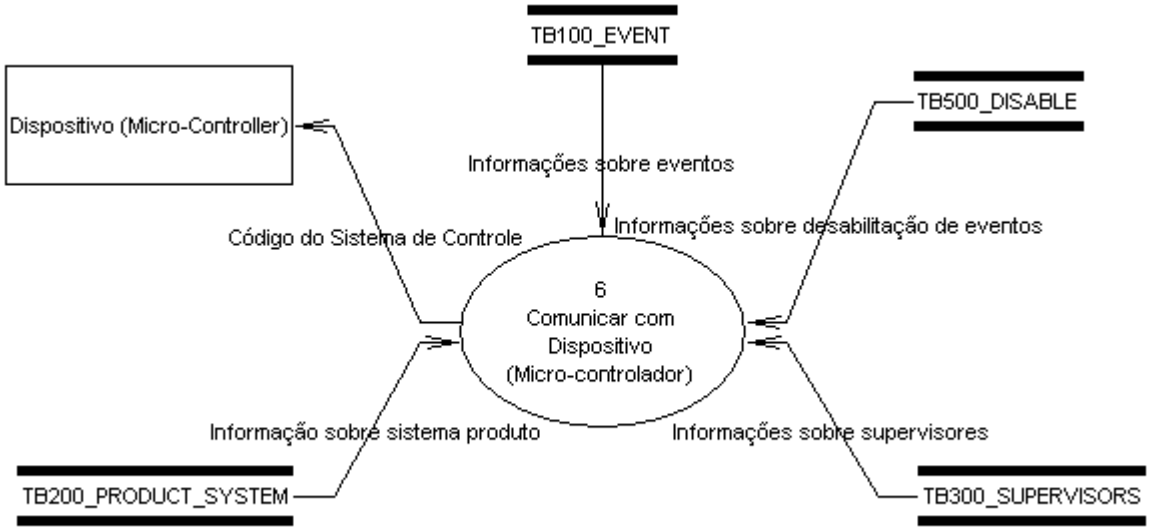


**Figura 4.12** Estrutura da tela para Comunicar com Dispositivos (Micro-Controller).

## 4.4.Código de Implementação do SC

Considera que a implementação ou realização do SC através da ferramenta ocorrerá de acordo com a concepção verificada por Costa et all (2005), que sugere a implementação do modelo Queiroz e Cury da seguinte forma:

- Supervisores e Sistema Produto em um dispositivo de baixo custo como por exemplo o microcontrolador;
- Seqüências Operacionais em um dispositivo industrial tal como Controlador Lógico Programável CLP.

Sendo assim a implementação do SC através da ferramenta Code Generator ocorrerá apenas no nível ods Supervisores Modulares e Sistema Produto, ficando a cargo do projetista a implementação do nível das Seqüências Operacionais em um dispositivo industrial como por exemplo o CLP.

## 4.5.Considerações sobre o Processo de Desenvolvimento

A tese proposta não foca na elaboração de uma metodologia de desenvolvimento de sistemas lógicos industriais, mas sim em prover uma ferramenta

de apoio que possa ser usada por projetistas seguindo alguns aspectos necessários para a elaboração de tal projeto com o uso da ferramenta. Outro ponto de suma importância é o fato de que a própria ferramenta em si já impõe um desenvolvimento sistemático, mas não obrigatório, como verificado anteriormente, é possível eliminar etapas do processo de desenvolvimento caso seja a vontade do projetista ou a sua confiança de um SC dito como ótimo.

De acordo com a filosofia seguida, a validação do comportamento do modelo gerado pode ser feita através da simulação, permitindo uma maior compreensão por parte do projetista do domínio do problema. É possível interpretar este modelo de simulação como uma representação de objetos, idéias ou sistema, o qual reflete os mesmos aspectos de comportamentos presentes no Sistema Real. Com essa abstração é possível validar o modelo obtido bem como o SC gerado pelo sistema, sendo assim é recomendado a execussão dessa etapa.

# 5
# Exemplo de Aplicação da Ferramenta Code Generator

Para exemplificação da metodologia proposta bem como do uso da ferramenta Code Generator, segue-se uma exemplificação do uso da ferramenta através da implementação de um sistema físico, será verificada aqui as etapas do ciclo de desenvolvimento – modelagem, síntese e implementação – bem como a sua inserçãoi na ferramenta case denominada de Code Generator e formas e técnicas de obtenção do SC através do desse software.

## 5.1.Descrição da Bancada de Teste (protótipo do sistema de manufatura)

Para a validação do software desenvolvido será adotado um equipamento didático fabricado pela FESTO, dividido em 6 (seis) módulos e detalhados a posteriori deste, onde cada módulo é composto por atuadores pneumáticos e elétricos, bem como sensores. Tal sistema simula uma linha de produção em que peças brutas são alimentadas em um buffer de entrada. Testadas: quanto à cor, material e espessura; furadas; testadas; e por fim armazenadas conforme a condição da peça ou cor da mesma, na figura a seguir observa-se a planta referida planta e denominada de MPS.

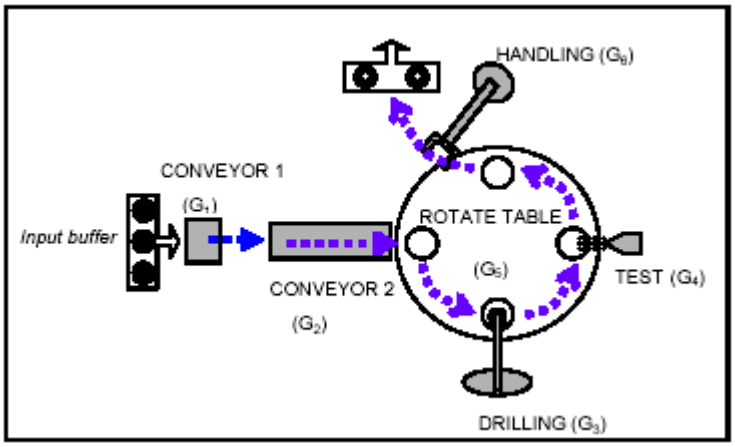


**Figura 5.1** Sistema MPS.

Para a identificação da peça o sistema conta com um conjunto de sensores: indutivo, capacitivo e óptico sendo assim o mesmo tem a capacidade de identificar os 3 (três) tipos básicos de peças encontrados no sistema para serem produzidos. Com a figura a seguir visualiza-se a distribuição proposta para o referido sistema a ser controlado.

O sistema é composto de seis subsistemas: fornecimento de material (G1), classificação e medição (G2), transporte (G5), processamento 1 (G3), processamento 2 (G4) e armazenamento (G6). O subsistema G1 tem por objetivo armazenar e suprir matéria prima sem classificação ao subsistema G2. O subsistema G2 realiza duas atividades sobre a matéria prima: classificando-a quanto ao tipo (cor e material) e realiza a medição da altura. A atividade de medição se dá em função de uma eventual não uniformidade da matéria prima, acarretando a existência de diferentes classes de tolerância dimensional. Faz-se necessário um dispositivo de medição da dimensão, sendo o próprio subsistema G2 capaz de descartar materiais que não se enquadram na tolerância desejada. Após realizadas estas operações, a matéria prima segue para o subsistema G5.

O subsistema G5 realiza o transporte entre os subsistemas G2, G3, G4 e G6. Os subsistemas G3 e G4 realizam processos de fabricação específicos. Por fim, o subsistema G6 armazena o produto final de acordo com os atributos obtidos no subsistema G2 (medição e classificação).

## 5.2.Etapa de modelagem

De acordo com a metodologia proposta, a primeira etapa (modelagem) consiste em representar em autômatos os subsistemas que compõem o sistema real e o conjunto de especificações (aplicação). Para tanto, poderá ser utilizada uma biblioteca de modelos conforme previsto no capítulo 3 a utilização de AMTs para tal fim. Para a modelagem dos subsistemas pode ser selecionado o modelo de dois estados (estado inativo e estado ativo), apresentado na figura 5.2.

$G_i, i = 1,\ldots,6$

$G_1$ : Atuador linear, atuador rotativo, vácuo;

$G_2$ : Elevador e esteira;

$G_3$ : Furadeira, atuador linear de fixação da peça;

$G_4$ : Atuador linear de teste;

$G_5$ : Mesa;

$G_6$ : Manipulador de três eixos.

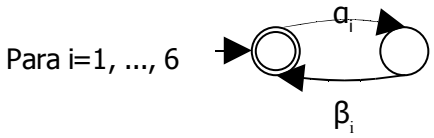


**Figura 5.2** Modelagem das sub-plantas do sistema real.

## 5.3. Especificações e supervisores

Para a modelagem das especificações analisa-se a configuração física do sistema, os roteiros de produção e as restrições de coordenação, de forma que o fluxo de matériais entre os subsistemas sejam representados corretamente. Para o correto fluxo de matéria entre os subsistemas G1 e G2 pode ser selecionada uma especificação de exclusão mútua entre os mesmos. Esta especificação impõe que G1 e G2 não podem funcionar simultaneamente e ao mesmo tempo define a sequência de execução das atividades relacionadas a estes subsistemas. Após a realização da classificação e medição pelo subsistema G2, a matéria prima segue para o subsistema de transporte G5. Este é definido como um transportador síncrono (mesa giratória) (GROOVER, 2001). Tal subsistema, movendo um produto

de uma posição a outra, acarreta na movimentação dos demais produtos para as posições subseqüentes. Neste momento, o projetista seleciona modelos considerando as seguintes informações:

i. Transporte síncrono de quatro posições;

ii. Primeira posição para chegada de peças, segunda e terceira posições para processos e quarta posição para retirada de peças.

iii. O conjunto de especificações relacionado ao transportador síncrono impõe o correto fluxo de matéria entre os subsistemas G2, G3, G4 e G6.

Os modelos das nove especificações necessárias ($E_a$, $E_b$, $E_{c1}$, $E_{c2}$, $E_{c3}$, $E_{c4}$, $E_{d1}$, $E_{d2}$, $E_{d3}$) para a aplicação inicial são apresentados na figura 5.3, onde verifica-se que:

- $E_a$ corresponde ao funcionamento do primeiro e do segundo sub-sistema, onde só deverá executar o segundo sub-sistema quando estiver uma peça a ser identinficada e transportada para o sistema de tranporte;
- $E_b$, corresponde ao funcionamento do sistema de transporte, onde somente deverá ser acionado mediante ao termino e todas os sub-sistemas que interage com tal sistema;
- $E_{c1}$, até $E_{c4}$: corresponde a especificação para o funcionamento do sistema de transporte sincrono, no caso o sistema só poderá ser executado quando todos os outros sistemas concluírem suas tarefas, se e somente se forem inicilizados.
- $E_{d1}$, até $E_{d3}$: Diz respeito as especificação para o funcionamento do sistema de transporte sincrono, e que somente poderá ser executao se um dos modulos que o aciona for executado e concluída sua tarefa.

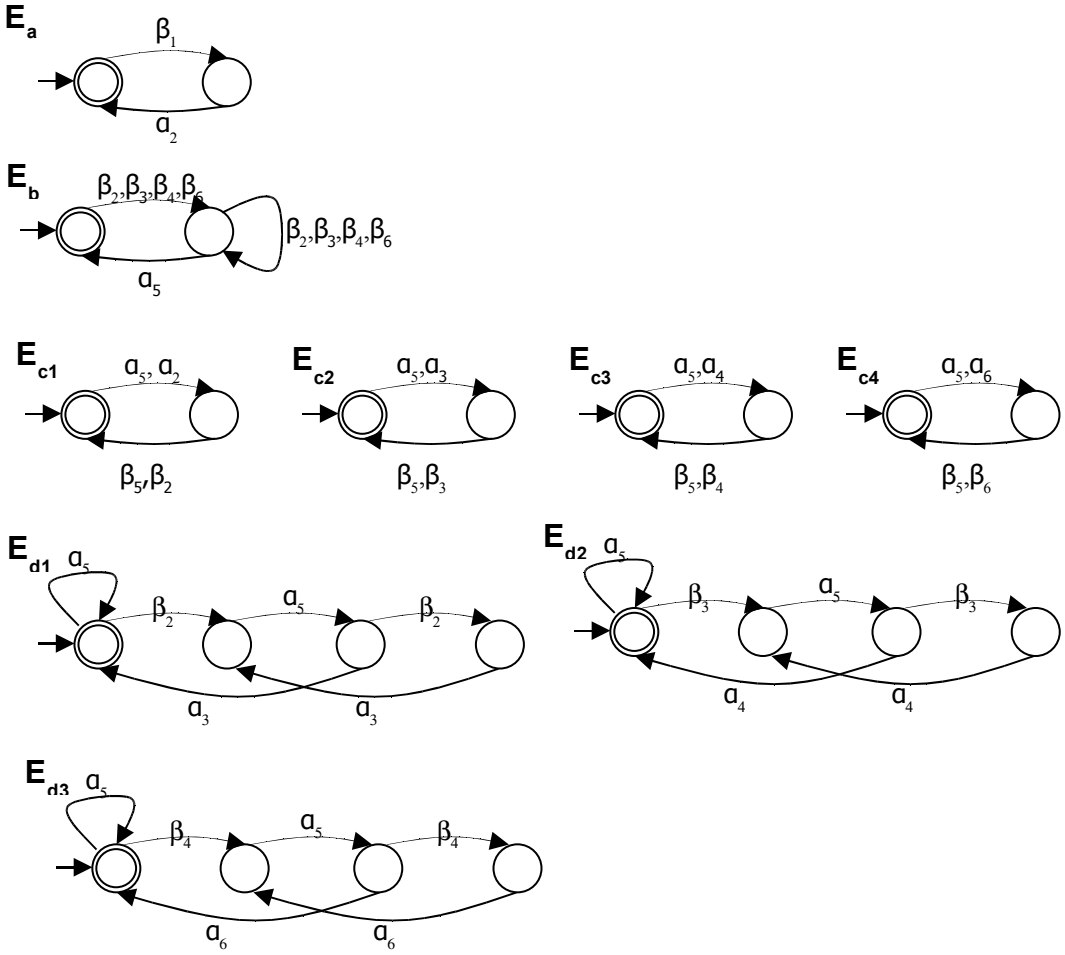


**Figura 5.3** Modelos das especificações para a aplicação inicial.

## 5.4.Etapa de síntese dos supervisores modulares

A próxima etapa – síntese – corresponde à aplicação da TCS e da abordagem modular local. Assim, para cada especificação selecionada (aplicação inicial e nova aplicação), um supervisor é obtido utilizando a ferramenta GRAIL. A ação conjunta dos supervisores obtidos restringe o comportamento dos subsistemas (que compõem o sistema real) às respectivas aplicações.

Nove supervisores são sintetizados, cada um com uma finalidade específica, a partir das especificações selecionadas e apresentadas na figura 5.5. Por exemplo, considera-se a especificação $E_{d1}$, a planta local é obtida através da composição síncrona dos autômatos correspondentes aos subsistemas que compartilham eventos com esta especificação ($G_{loc,d1} = G_2 \,||\, G_3 \,||\, G_4$). A especificação local é obtida através da composição da especificação genérica $E_{d1}$ com a correspondente planta

local ($E_{loc,d1}$= $E_{d1}$ || $G_{loc,d1}$). Pode-se então calcular a máxima linguagem controlável contida na especificação, que é SupC ($E_{loc,d1}$, $G_{loc,d1}$). Em seguida, através de um algoritmo de minimização de supervisores (VAZ & WONHAM, 1986), obtém-se um supervisor com um menor número de estados e com a mesma ação de controle. A tabela 2 apresenta os supervisores calculados a partir dos modelos dos subsistemas e das especificações selecionados para a aplicação inicial.

| Especificação genérica | Planta Local | Especificação local | Máxima linguagem controlável (Supervisor Modular) |
|---|---|---|---|
| $E_a$ | $G_{loc,a} = G1 \parallel G2$ | $E_{loc,a} = G_{loc,a} \parallel E_a$ | $SupC(G_{loc,a}, E_{loc,a})$ |
| $E_b$ | $G_{loc,b} = G2 \parallel G3 \parallel G4 \parallel G5$ | $E_{loc,b} = G_{loc,b} \parallel E_b$ | $SupC(G_{loc,b}, E_{loc,b})$ |
| $E_{c1}$ | $G_{loc,c1} = G2 \parallel G3$ | $E_{loc,c1} = G_{loc,c1} \parallel E_{c1}$ | $SupC(G_{loc,c1}, E_{loc,c1})$ |
| $E_{c2}$ | $G_{loc,c2} = G3 \parallel G4$ | $E_{loc,c2} = G_{loc,c2} \parallel E_{c2}$ | $SupC(G_{loc,c2}, E_{loc,c2})$ |
| $E_{c3}$ | $G_{loc,c3} = G3 \parallel G5$ | $E_{loc,c3} = G_{loc,c3} \parallel E_{c3}$ | $SupC(G_{loc,c3}, E_{loc,c3})$ |
| $E_{c4}$ | $G_{loc,c4} = G3 \parallel G6$ | $E_{loc,c4} = G_{loc,c4} \parallel E_{c4}$ | $SupC(G_{loc,c4}, E_{loc,c4})$ |
| $E_{d1}$ | $G_{loc,d1} = G2 \parallel G3 \parallel G4$ | $E_{loc,d1} = G_{loc,d1} \parallel E_{d1}$ | $SupC(G_{loc,d1}, E_{loc,d1})$ |
| $E_{d2}$ | $G_{loc,d2} = G3 \parallel G4 \parallel G5$ | $E_{loc,d2} = G_{loc,d2} \parallel E_{d2}$ | $SupC(G_{loc,d2}, E_{loc,d2})$ |
| $E_{d3}$ | $G_{loc,d3} = G3 \parallel G5 \parallel G6$ | $E_{loc,d3} = G_{loc,d3} \parallel E_{d3}$ | $SupC(G_{loc,d3}, E_{loc,d3})$ |

Tabela 5.1 Modelos utilizados no procedimento de síntese e supervisores locais resultantes.

Por exemplo, a figura 5.4 apresenta o supervisor SupC($G_{loc,d1}$, $E_{loc,d1}$), resultante da especificação $E_{d1}$. A linha tracejada vermelha indica a ação de controle do supervisor, que é desabilitar eventos controláveis dos subsistemas G2, G5 e G3.

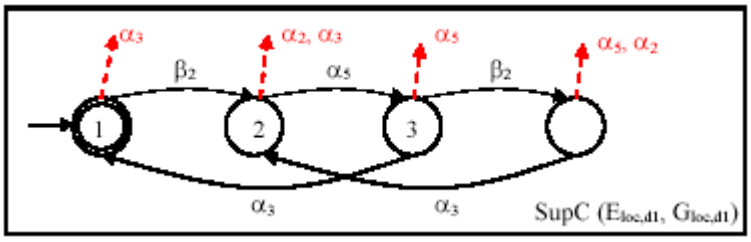


**Figura 5.4** Exemplo de supervisores.

## 5.5.Cadastro do Sistema Produto e Supervisores Modulares

O uso da ferramenta Code Generator começa a partir da conclusão das 2 primeiras etapas: Modelagem e Síntese. Devendo os modelos formais dos subsistemas e dos supervisores encontrados serem incluídos na ferramenta. Para exemplificação de implementação e codificação do modelo acima no software Code Generator, considera-se a mesa $G_5$ , onde para tal tem-se a seguinte codificação verificada através da figura 5.5:

```
(START) |- 0
0 a5 1
1 b5 0
0 -| (FINAL)
```

**Figura 5.5** Codificação do autômato na linguagem reconhecida pelo GRAIL.

Deve-se Inserir no software o número de Sub-Sistemas, no caso exemplificado a seguir com a figura serão 6 (seis), e clicar em <*Create*> e inserir os códigos e compilá-los afim de carregar as informações nas tabelas conforme verificado no capítulo anterior através dos seu DFD. A seguir visualiza-se um esboço da tela inserindo um código reconhecido pelo sistema afim de interpretar um automato.


**Figura 5.6** Tela do Sistema para inserção do Sistema Produto.

Para exemplificar a implementação e codificação dos supervisores acima relatados no software Code Generator, considera-se a figura 5.7, deve-se Inserir no software o número de Supervisores, no caso exemplificado a seguir com a figura serão 9 (nove), e clicar em <*Create*> e inserir os códigos e compilá-los afim de carregar as informações nas tabelas conforme verificado no capítulo anterior através dos seu DFD. A seguir visualiza-se um esboço da tela inserindo um código reconhecido pelo sistema afim de interpretar um automato.

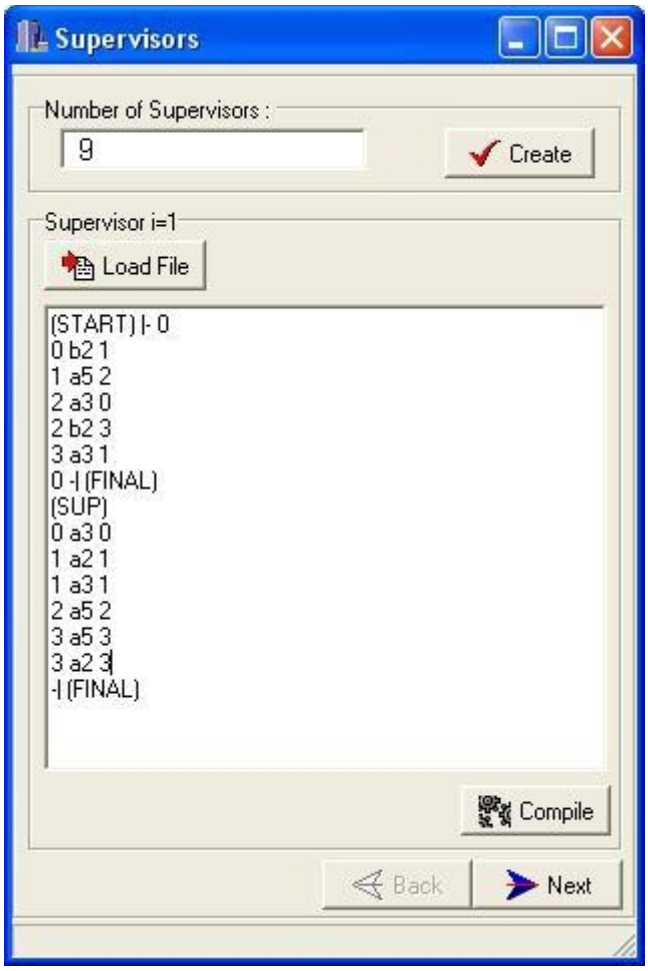


**Figura 5.7** Tela do Sistema para inserção dos Supervisores.

## 5.6.Características da Estrutura de Controle

A ação de controle consiste em desabilitar eventos controláveis $\alpha_i$ dos módulos $G_i$ . Caso nenhum supervisor desabilite um determinado momento um evento $\alpha_i$ , este é gerado imediatamente pelo programa. Em conjunto com os eventos $\alpha_i$ são gerados comandos e_ini_SOk, que inicializam as Sequências Operacionais k.

Ao final do programa de execução de uma sequência operacional SOk, é gerada uma resposta e_fim_SOk, que por sua vez é condição para a geração de eventos $\beta_i$ . Eventos $\alpha_i$ e $\beta_i$ atualizam os supervisores modulares. Na figura a seguir segue a estrutura sugerida de implementação do SC concebido pelo software Code Generator.

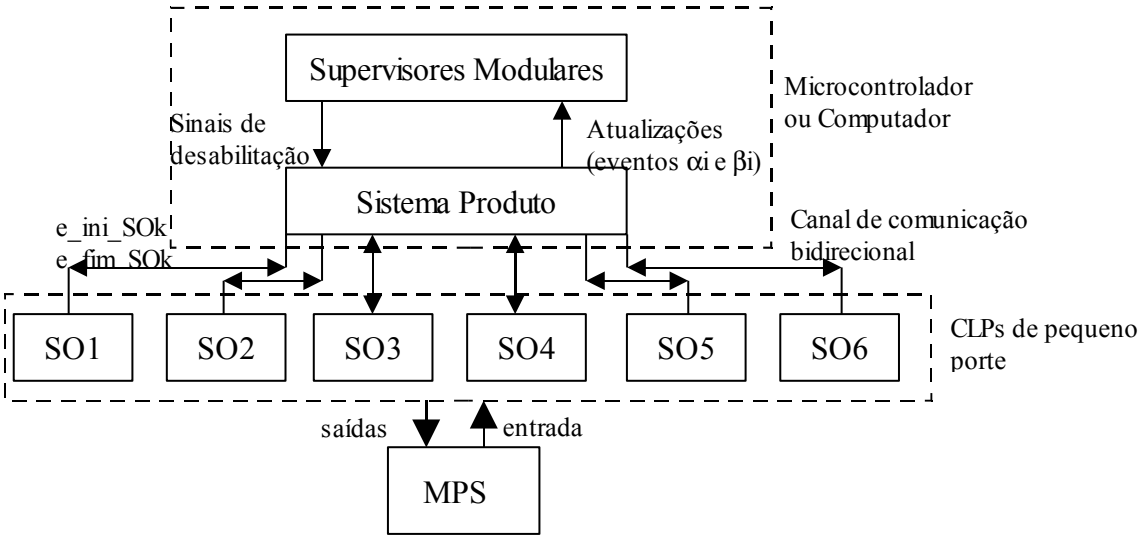


**Figura 5.8** Estrutura do SC sugerido para implementação.

### 5.6.1.Validação

Tal realização ocorrerá através de um servidor e um cliente do tipo Socket no qual refere-se a comunicação ethernet, ou seja, através da intranet um software de simulação poderá comunicar com o Code Generator e assim realizar a validação do SC através da simulação, conforme sugerido e discutido no capítulo 3 a respeito da validação do SC através de simulação do sistema. A validação proposta aqui ocorrerá de acordo com o reconhecimento da linguagem gerada $L(G)$ e pela correta produção de um determinado bem em um meio virtual. Para a primeira validação a da linguagem gerada $L(G)$ observa um exemplo de reconhecimento com a tela do sistema Code Generator onde o projetista poderá vizualizar tal linguagem. Para a segunda validação, a respeito da produção, deverá ser criado uma interface em um programa de simulação, conforme vizualizado na figura 5.10 do presente onde foi simulado a produção do sistema MPS no software Em-Plant.

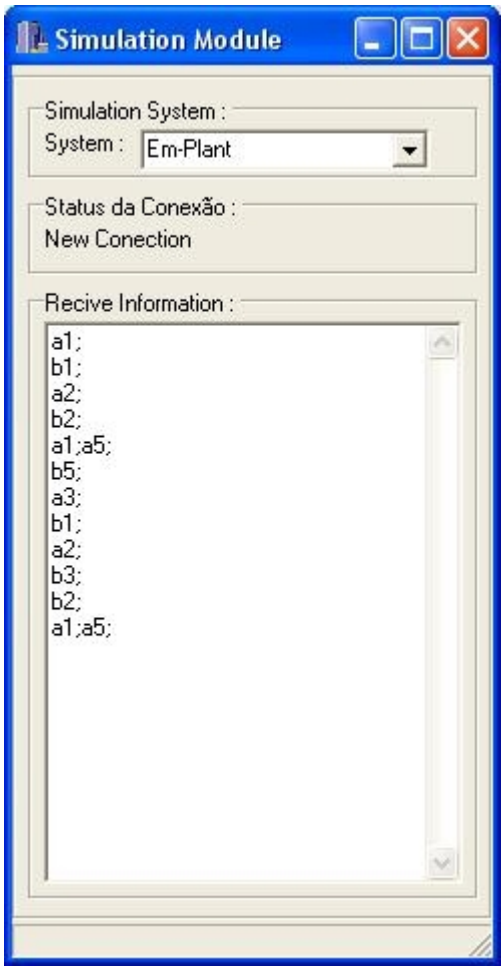


**Figura 5.9** Tela referente a realização do sistema de simulação.

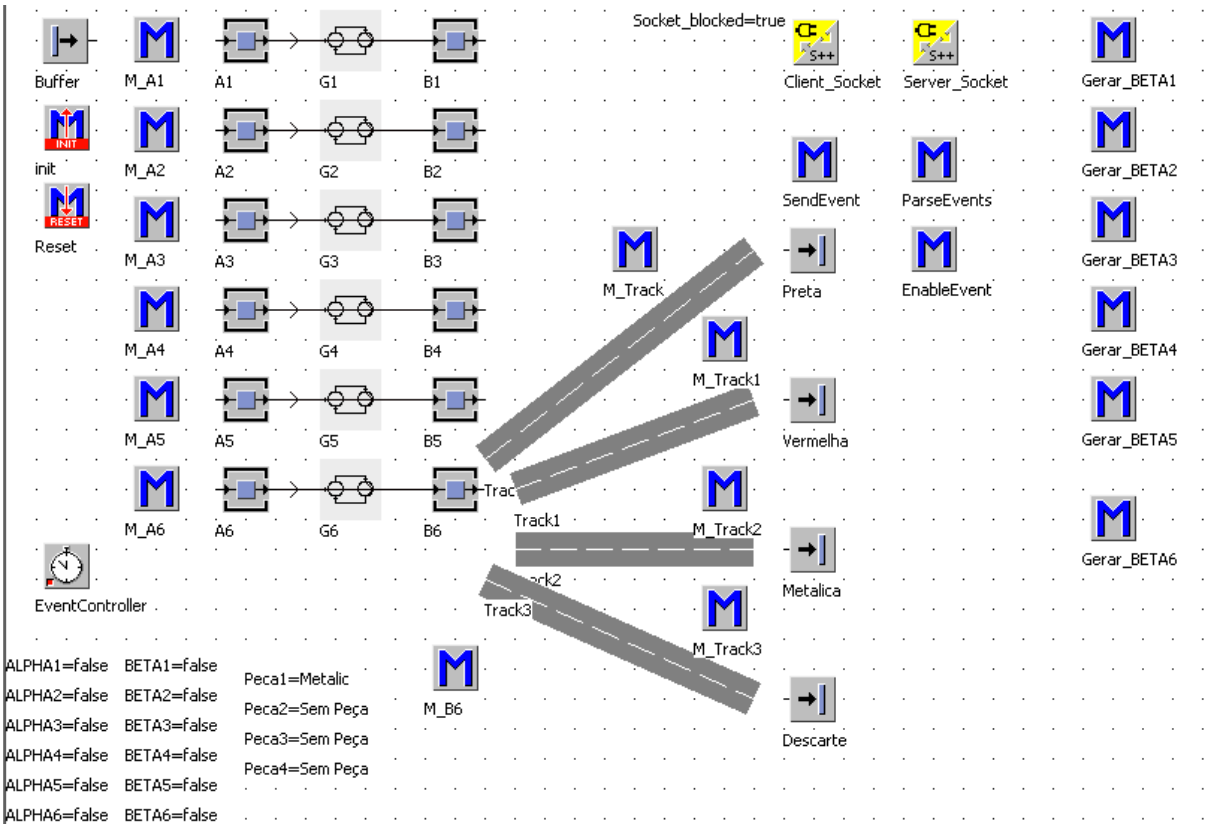


**Figura 5.10** Tela da Simulação do Sistema no Em-Plant.

A comunicação estre os 2 software ocorrerá perante a seguinte regra de consepção:

i. O servidor do software Code Generator espera um sinal de inicialização do software de simulação, esse sinal inicialmente foi implementado como sendo a plavra *"x;";*

ii. Concluída tal tarefa o software Code Generate deverá enviar os *comandos* para o software de simulação, a fim de desempenhar determinada rotina de trabalho;

iii. Ao termino do trabalho o software de simulação deverá informar as respostas, conforme projetado pelo SC e assim sucessivamente, lembrando sempre que toda vez que o software Code Generator receber o sinal de inicialização o mesmo irá inicializar o SC.

### 5.6.2.Realização Física

A comunicação dos dispositivos industriais (nível das sequências operacionais) com o microcontrolador pode ser estabelecida através de entradas e saídas físicas. Nesse caso, deve-se prever uma saída e uma entrada para cada CLP (e_fim_SOk e e_ini_SOk, respectivamente) e também k saídas e k entradas para o microcontrolador (e_ini_SOk e e_fim_SOk (k = 1,..., 6), respectivamente).

Alternativamente, pode-se estabelecer uma rede de comunicação entre esses dispositivos industriais onde estão implementadas as sequências operacionais. A desvantagem nesse caso reside na dependência de protocolos proprietários. O usuário deverá, nesse caso, ter o conhecimento das regras de comunicação de cada dispositivo utilizado. No caso tratado, o microcontrolador deve enviar o pacote de informações $\Psi$ (conjunto e_ini_SOk) e receber o pacote de informações $\Theta$ (conjunto e_fim_SOk). É necessário, entretanto, que se tenha conhecimento prévio de como cada fabricante envia e recebe dados. Deve-se portanto buscar maneiras para entender o protocolo de cada fabricante.

A realização física ocorrerá através da implementação por computador ou através do micro-controlador, sendo que o ultimo o sistema gera o código em Assembler para micro-controlador PIC. O código em questão está de acordo com o algoritmo proposto e relatado por Costa et all (2005) e visualizado a seguir através da figura 5.11.

Ainda com relação a figura 5.11, percebe-se o critério utilizado para atualizar os estados dos supervisores, no caso primeiro verifica-se a ocorrência de eventos não controláveis, para a mudança de estados, logo em seguida deve-se verificar qual estados deverão ser habilitados e liberar o sinal de comando para os diversos dispositivos industriais. Essa regra tem como objetivo principal garantir a correta transição de estados dos supervisores.

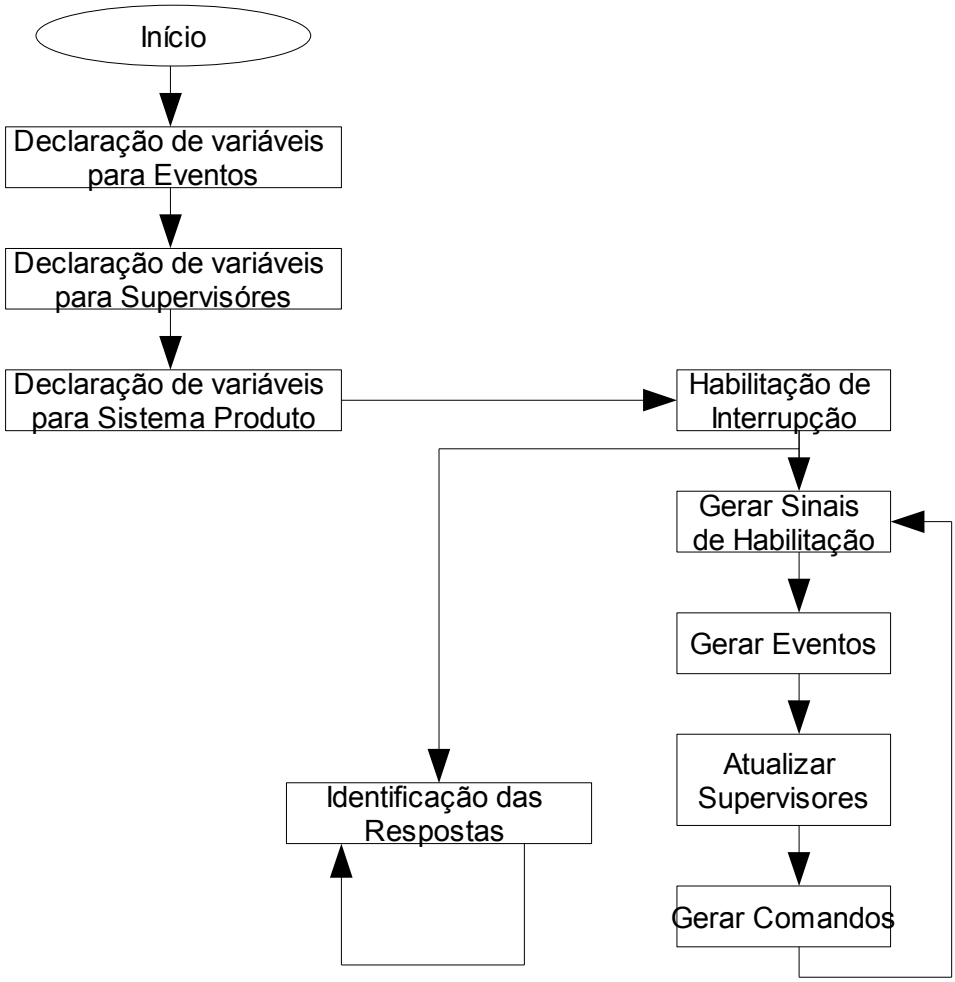


**Figura 5.11** Fluxograma do Sistema de Controle.

Observando o algorítimo proposto acima percebe-se que inicalmente deve-se definir algumas variáveis de controle dos eventos, supervisores e sistema produto, conforme exemplificação a seguir:

```
;***** Definição de Variáveis
  CONTROLABLE       EQU     0x22    ; variable used for states
  UNCONTROLABLE     EQU     0x23    ; variable used for states disable

  SUP1              EQU     0x24    ;variable used for Supervisor 1
  ...

  G1                EQU     0x34    ;variable used for Model 1.
  ...

  W_TEMP            EQU     0x40    ; variable used to save W Register
  STATUS_TEMP       EQU     0x41    ; variable used to save STATUS
Register
```

Finalizada a etapa de declaração de variáveis usado pelo SC, deve-se habilitar o uso de interrupção do micro-controlador, conforme exposto no algorítimo da figura 5.8, essa rotina servirá para identificar todo e qualquer sinal referente a *"Resposta"* proveniente de um dispositivo industrial, a seguir segue o código para a realização dessa tarefa.

```
;*****************************************************************
;* Salva Informações dos Registradores W e STATUS                *
;*****************************************************************
                MOVWF   W_TEMP
                SWAPF   STATUS, W
                MOVWF   STATUS_TEMP

;*****************************************************************
;* Desliga o FLAG da Interrupção                                 *
;*****************************************************************
                BCF             INTCON, INTF

;*****************************************************************
;* Tarefa da Interrupção                                         *
;*****************************************************************
                BANKSEL PORTB

                MOVF    PORTB, w
                MOVWF   UNCONTROLABLE

;*****************************************************************
;* Verificar sinais não controláveis                             *
;*****************************************************************
TRANSICAO_ESTADO
                BTFSC   UNCONTROLABLE,  1
                CALL    b1
                BTFSC   UNCONTROLABLE,  2
                CALL    b2
               ...
               ...
               ...

RETORNA:
;*****************************************************************
;* Restaurar Registradores W e Status                            *
;*****************************************************************
                SWAPF   STATUS_TEMP,    W
                MOVWF   STATUS
                SWAPF   W_TEMP,         F
                SWAPF   W_TEMP,         W
                RETFIE
```

Para o código exemplificado anteriormente verifica-se e destaca-se a chamada de funções de identificação da ocorrência da *"Resposta"* , no caso a função b1, exemplificada a seguir através da codificação da mesma.

```
;*****************************************************************
;* Atualizar os estados dos supervisores com a ocorrência b1     *
;*****************************************************************
B1
                ;*************************************************
                ;* Muda estado do Modelo 1 (Maquina 1)           *
                ;*************************************************
                MOVLW   .2
                XORWF   G1

                BTFSC   STATUS, Z
                MOVLW   .1
                MOVF    G1,     w
                MOVWF   G1
                ;*************************************************
                ;* Muda estado do Supervisor SUP1 com a ocorrência b1
                ;*************************************************
                MOVLW   .1
                XORWF   SUP1

                BTFSC   STATUS, Z
                MOVLW   .2
                MOVF    SUP1,   w
                MOVWF   SUP1
                ;*************************************************
                ;* Zerar BIT 1 (IDENTIFICAÇÃO REALIZADA)         *
                ;*************************************************
                BCF             UNCONTROLABLE,  1

                RETURN
```

Concluída a etapa referente a identificação dos sinais provenientes das seqüências operacionais e atualizado o estados de todos os supervisores e do sistema produto com os referidos sinais, chega-se a hora de enviar a camada referente ao sistema produto do SC os sinais de habilitação para que possa ser gerado os "*Comandos*" para as seqüências operacionais, a seguindo-se tal codificação.

```
;*****************************************************************
;* Transição de estados dos supervisores                         *
;*****************************************************************
CARREGA_ALPHA
                ;*************************************************
                ;* Verificar estado 1 do SUP1                    *
                ;*************************************************
                MOVLW   .1
                XORWF   SUP1

                BTFSC   STATUS, Z
                CALL    PS1_1

                ;*************************************************
```

```
;* Verificar estado 2 do SUP1                          *
;*****************************************************
MOVLW   .2
XORWF   SUP1

BTFSC   STATUS, Z
CALL    PS2_1
...
...
...
```

Para o código exemplificado anteriormente verifica-se e destaca-se a chamada de funções para gerar eventos, no caso a função PS1_1, exemplificada a seguir através da codificação da mesma.

```
;*****************************************************************
;*GERAR EVENTO                                                   *
;*****************************************************************
PS1_1
        MOVLW   .1
        XORWF   G1

        BTFSC   STATUS, Z
        BCF     CONTROLABLE,    1
        NOP

        RETURN
```

Após a conclusão dessa etapa deverá criar e enviar os sinais as seqüências operacionais, sendo assim segue o código referente a esta tarefa.

```
;*********************************************************
;* Gerar comando 1                                       *
;*********************************************************
BTFSC   CONTROLABLE,    0
BSF             PORTC,  0
NOP

BTFSC   CONTROLABLE,    0
CALL    a1
NOP
...
...
...
```

Para o código exemplificado anteriormente verifica-se e destaca-se a chamada de funções para atualização dos supervisores conforme ocorrência do

comando, no caso a função a1, exemplificada a seguir através da codificação da mesma.

```
;*****************************************************************
;* Atualizar os estados dos supervisores com a ocorrência a1     *
;*****************************************************************
a1
                ;*************************************************
                ;* Muda estado do Modelo 1 (Maquina 1)           *
                ;*************************************************
                MOVLW   .1
                XORWF   G1

                BTFSC   STATUS, Z
                CALL    SEND_2
                MOVF    G1,     w

                MOVWF   G1

                RETURN
```

## 5.7.Interface de Comunicação

Duas condições devem ser obedecidas na implementação da estrutura de controle:

i. A evolução do sistema produto através de um evento controlável só pode ocorrer se todos os supervisores locais podem exercer ação de desabilitação sobre este evento estiverem atualizados e se nenhum destes supervisores estiver desabilitando este evento (garante a ação conjunta dos supervisores);
ii. Em cada um dos supervisores só é permitida a evolução de estado através de um único evento por ciclo de atualização (garante a correta evolução dos supervisores para o atual modelo de implementação dos mesmos e a conformidade com a teoria de controle supervisório, a qual não prevê a ocorrência simultânea de eventos).

Para atender as duas condições previstas anteiormente, deve-se criar uma interface de comunicação, conforme relatado por Costa et all (2005) e mostrado na figura 5.12, para a comunicação de um dispositivo de baixo custo, quer seja um micro-controlador quer seja um computador, faz-se necessário a implementação de

uma interface de comunicação que deverá garantir os seguintes aspectos, além dos outros dois citados anteriormente:

- Correto reconhecimento entre o dispositivo de baixo custo e o dispositivo industrial;
- Equiparação de tensões para os 2 dois tipos de dispositivos, haja visto que o dispositivo de baixo custo observado nesta pesquisa necessita de uma diferença de potencial de 5V e o dispositivo industrial uma diferença de potencial de 24V.

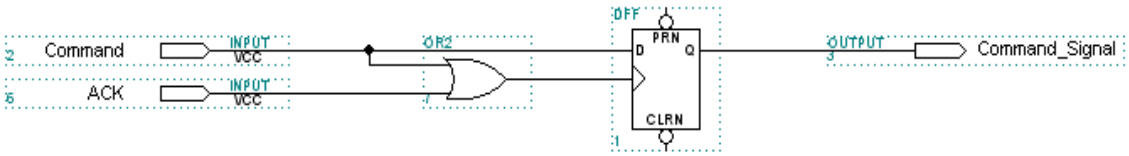


**Figura 5.12** Exemplo de um circuito para interface de comunicação.

A seguir visualiza-se através da figura 5.13 os sinais de simulação do circuito exemplificado anteriormente, onde constata-se que o sinal do micro-controlador é mais rápido que o sinal de reconhecimento do dispositivo industrial sendo assim o sinal se manterá até a geração de um sinal de reconhecimento do dispositivo industrial.

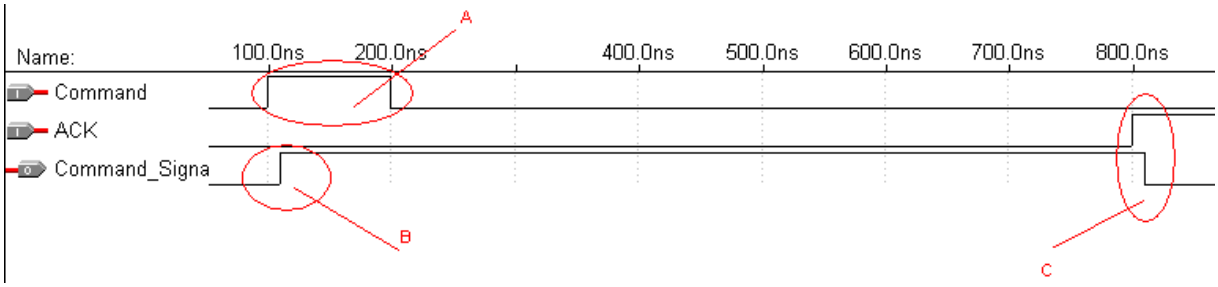


**Figura 5.13** Simulação de sinais para a interface de comunicação.

## 5.8. Resumo do capítulo

Nesse capítulo verificaram-se aspectos relacionados aos teste e resultados obtidos com a utilização do software Code Generator, tais testes basearam-se na planta MPS. Outro aspecto discutido aqui é as várias formas de realização e qual a forma de comportamento do sistema.

# 6
# Conclusão e Trabalhos Futuros

O trabalho apresentou uma metodologia para o projeto de sistemas automatizados e integrados de manufatura bem como uma ferramenta case para a utilização dessa metodologia proposta. A inserção da TCS baseada na teoria de Linguagens e Autômatos, de ferramentas computacionais para a síntese e simulação, de biblioteca de modelos de subsistemas e especificações trouxe uma diminuição do tempo de desenvolvimento do sistema automatizado de manufatura.

A TCS é uma abordagem formal que permite a síntese automática de supervisores ótimos. Aliada a TCS, a abordagem modular traz uma maior agilidade ao projeto. Vantagens da abordagem modular local. A biblioteca de modelos, além de facilitar e sistematizar a etapa de modelagem, permite a reutilização de modelos em projetos subseqüentes. A integração das etapas de simulação e implementação permitiu uma maior confiabilidade na validação, otimização e realização da estrutura de controle.

A metodologia proposta apresenta ainda algumas limitações que pressupõem a continuidade do trabalho desenvolvido. No início do ciclo, devem ser aprofundados os critérios de segmentação do sistema real em subsistemas, bem como a identificação de um conjunto de especificações para uma dada aplicação.

As bibliotecas de modelos de subsistemas e de especificações devem ser constantemente atualizadas. A experiência de cada novo projeto deve ser utilizada na atualização das bibliotecas. Entretanto, a seleção dos modelos ainda é muito dependente da experiência do projetista. Devem ser pesquisados métodos para

sistematizar a seleção de modelos da biblioteca a partir do sistema real e da aplicação.

Como resultado dessa abordagem metodológica aliado ao desenvolvimento da ferramenta em questão têm-se os elementos necessários para a documentação detalhada do projeto técnico. Para tal fim, existem ainda outros fatores a serem considerados, tais como manutenção, relações comerciais, custos, atualizações e revisões técnicas de equipamentos já instalados, dentre outros.

A biblioteca das Tecnologias Avançadas de Manufatura – AMTs deve também ser constantemente revisada e atualizada. Uma abordagem sistemática deve ser pesquisada para auxiliar tanto no reconhecimento dos subsistemas a partir das tecnologias, quanto na definição das tecnologias de controle e comunicação necessárias a implementação e realização da estrutura de controle modular.

## Referências Bibliográficas

AKESSON, K. (2002) - Methods and Tools in Supervisory Control Theory. Thesis For The Degree Of Doctor Of Philosophy, Chalmers University of Technology, Sweden.

BALEMI, S., HOFFMANN, G. J. and GYUGYI, P. et al (1993). Supervisory control of a rapid thermal multiprocessor. IEEE Transactions on Automatic Control, 38(7), 1040-1059.

CARROL, J.; LONG, D. Theory of finite automata. Prentice-Hall International Editions, 1989.

CASSANDRAS C. G. & LAFORTUNE S., *Introduction to discrete event systems*, Kluwer Academic Publishers, 1999.

CHANDRA, V., ORUGANTI, B. and KUMAR, R. (2002). UKDES: A graphical software tool for the design, analysis & control of discrete event systems. clue.eng.iastate.edu/~rkumar/PUBS/ukdes.ps.17:34.

COSTA, G. O.; PORTELA, E. A.; BUSETTI, M. A. (2005). COST BASED IMPLEMENTATION OF MODULAR SUPERVISORY CONTROL THEORY, In: 16th IFAC World Congress, Praga.

CURY, J. E. R. (2001). Teoria de Controle Supervisório de Sistemas a Eventos Discretos, V Simpósio Brasileiro de Automação Inteligente, Gramado, RS.

FABIAN, M. & A. HELLGREN (2000). Desco – a Tool for Education and Control of Discrete Event Systems. Kluwer Academic Publishers.

GOUVÊA DA COSTA, S.; PLATTS, K.; FLEURY, A. Advanced Manufacturing Technology: defining the object and positioning it as an element of manufacturing strategy. In: VI Internationa Conference on Industrial Engineering and Operations Management – VI ICIEOM (6. : 2000 : São Paulo). Proceedings. São Paulo, 2000.

GROOVER, M. Automation, Production Systems, and Computer-Integrated Manufacturing. 2a. ed. New Jersey : Prentice-Hall, 2001.

HOLLOWAY, L. E.; KROGH, B. H.; GIUA, A. A survey of Petri nets methods for controlled discrete event systems. Discrete Event Dynamic Systems: Theory and Applications, 7, pp. 151-190. Kluwer Academic Publishers, Boston, 1997.

IEC 61131-3 International Electrotechnical Commission (1998). Programmable Controllers. Programming Languages. International Eletrotechnical Comission., Preparation of function charts for control systems**,** IEC 848. Switzerland, 1988.

MOORE, P.R; PU J.; NG, H.C. et al. (2003) – Virtual engineering: an integrated approach to agile manufacturing machinery design and control. Mechatronics, 13, p. 1105–1121.

QUEIROZ, M. H. & CURY, J. E. R. (2002). Controle Supervisório Modular de Sistemas de Manufatura. Revista controle & automaçã, SBA, 13, pp.115-125.

QUEIROZ, M.H. & CURY, J.E.R. “Synthesis and implementation of local modular supervisory control for a manufacturing cell”, Discrete Event Systems: Analysis and Control, Kluwer Academic Publishers, pp. 103-110. (Proc. WODES 2002b).

RAMADGE P.J. & WONHAM W.M., “The control of discrete event system*” Proceedings IEEE, Special Issue on Discrete Event Dynamic Sys tems,* vol. 77, pp. 1202-1218, January 1989.

RAYMOND, D. & WOOD, D. (1996). Grail: Engineering automata in C++: Version 2.5, http://www.csd.uwo.ca/research/grail/ verificada em 31 de janeiro de 2003.

VAZ, A. F. & WONHAM, W. M. (1986). On supervisor reduction in discrete-event systems. International Journal of Control, 44(2):475-491.

VIEIRA, A. D. CONTRIBUIÇÕES À IMPLEMENTAÇÃO DE SISTEMAS DE CONTROLE SUPERVISÓRIO, Tese de doutorado, 2004.

# Livros Grátis

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

Milhares de Livros para Download:


Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

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