

NICOLE BEATRIZ PORTILLA CONSTAIN

**INTEGRAÇÃO DE SISTEMAS SCADA COM A
IMPLEMENTAÇÃO DE CONTROLE SUPERVISÓRIO EM CLP
PARA SISTEMAS DE MANUFATURA**

**FLORIANÓPOLIS
2011**

UNIVERSIDADE FEDERAL DE SANTA CATARINA

**PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS**

**INTEGRAÇÃO DE SISTEMAS SCADA COM A
IMPLEMENTAÇÃO DE CONTROLE SUPERVISÓRIO EM CLP
PARA SISTEMAS DE MANUFATURA**

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas e da Universidade Federal de Santa Catarina para a obtenção do Grau de Mestrado em Engenharia de Automação e Sistemas

Orientador: Prof. Dr. Max Hering de Queiroz

Nicole Beatriz Portilla Constain

Florianópolis, Setembro de 2011

Dedico este trabalho humildemente a Deus e seu filho Jesus Cristo, a meus professores, família e amigos por todos os ensinamentos, apoio e confiança que depositaram em mim para a realização desse trabalho, o que me impulsiona a continuar seguindo em frente.

AGRADECIMENTOS

Agradeço primeiramente a Deus pelas bênçãos da vida e capacidade intelectual, que me permitiram concluir este trabalho.

Aos meus amados pais Nicolás e Beatriz e minhas irmãs Sandra e Alejandra por seu amor incondicional e porque sempre me apoiaram e impulsionaram a continuar seguindo em frente, mesmo sob pressões e desgaste físico e emocional;

Ao meu orientador, Max Hering de Queiroz e meu co-orientador Jose Eduardo Ribeiro Cury pelo apoio, compreensão, paciência e conhecimentos transmitidos. Levo-os como exemplo para a minha nova etapa profissional.

Aos professores Carlos Barros Montez e Marcelo Stemmer pela instrução no âmbito prático da docência.

A todos os professores do DAS pela instrução que me capacitou a desenvolver este trabalho.

À Universidade Federal de Santa Catarina pela oportunidade de atuar como aluna bolsista do Programa de Pós-graduação. O meu respeito por esta grande e importante universidade.

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela concessão da bolsa de estudo que viabilizou o pleno desenvolvimento de meu Mestrado.

À todos os meus amigos que vivenciaram todas as dificuldades e sempre me incentivaram nesta longa e difícil jornada.

Enfim, agradeço a todos os envolvidos, que direta ou indiretamente, contribuíram para que eu pudesse alcançar mais este estágio em minha vida.

A todos, muito obrigada!

RESUMO

Os sistemas SCADA (*Supervisory Control and Data Acquisition*) são sistemas que utilizam tecnologias de computação e comunicação para automatizar a monitoração e o controle de processos industriais, efetuando coleta de dados, os quais são apresentados de modo amigável para o operador, através de interfaces homem-máquina (IHM). Dentre esses processos encontram-se os sistemas de manufatura, cuja dinâmica, sob a ótica do problema de coordenação, os enquadra na classe de Sistemas a Eventos Discretos (SED). A Teoria de Controle Supervisório (TCS) baseia-se em modelos de autômatos e linguagens para síntese de supervisores ótimos para SED que podem ser traduzidos em código estruturado para controladores lógicos programáveis (CLP). Este trabalho apresenta uma proposta de metodologia para o desenvolvimento integrado de sistemas SCADA com a programação de controle supervisório em CLP para sistemas de manufatura. A metodologia proposta está constituída de 8 fases: projeto informacional; síntese de controle supervisório seguindo a abordagem modular local da TCS; emulação da atuação dos supervisores na planta; implementação estruturada do controle supervisório em CLP; implementação de funcionalidades básicas do sistema SCADA; avaliação de funcionamento do sistema real; implementação de funcionalidades gerais do sistema SCADA e, por último, validação do sistema integrado. Esta metodologia é aplicada ao controle e supervisão de uma célula flexível de manufatura do LAI-UFSC. Os resultados obtidos demonstram sistematização, flexibilidade e eficiência na realização do projeto de controle e supervisão do sistema, além de permitir estruturação e validação do programa do CLP e do sistema SCADA.

Palavras-chave: sistemas SCADA; controle supervisório; controladores lógicos programáveis, sistemas a eventos discretos; sistemas de manufatura.

ABSTRACT

SCADA systems (Supervisory Control and Data Acquisition) are systems using computing and communication technologies to automate the monitoring and control of industrial processes, making data collection, which are presented in a friendly form for the operator, through Human Machine Interface (HMI). Among these processes are the manufacturing systems, whose dynamics, from the perspective of the coordination problem, include them in the class of discrete event systems (DES). The Supervisory Control Theory (SCT) is based on models of automata and languages for optimal synthesis of supervisors for DES, which can be translated into structured code for programmable logic controllers (PLC). This work presents a proposal of methodology for integrated development of SCADA systems with the programming of supervisory control in PLC for manufacturing systems. The proposed methodology is comprised of eight phases: informational design; synthesis of supervisory control, following the local modular approach of TCS; emulation of the performance of supervisors in the plant; structured implementation of supervisory control in the PLC; implementation of basic functionality of SCADA system, evaluation of operation of the real system; implementation of the overall functionality of the SCADA system and, finally, validation of the integrated system. This methodology is applied to the control and supervision of a flexible manufacturing cell of the LAI-UFSC. The results demonstrate systematic, flexibility and efficiency in carrying out the project control and supervision system, and allows structuring and validation of the program PLC and SCADA system.

Keywords: SCADA systems; supervisory control; programmable logic controllers; discrete event systems; manufacturing systems.

LISTA DE FIGURAS

Figura 1.1 Pirâmide da automação	1
Figura 2.1 Arquitetura típica de um sistema SCADA	8
Figura 2.2 Layout geral de um sistema SCADA	10
Figura 3.1 Visualização gráfica de um gerador	22
Figura 3.2 Estrutura em malha fechada do controle supervísório	24
Figura 3.3 Exemplificação para obtenção de supervisores locais	26
Figura 3.4 Célula flexível de manufatura (SILVA, 2010)	28
Figura 3.5 Geradores para os subsistemas da CFM (SILVA, 2010)	31
Figura 3.6 Especificações para a CFM (SILVA, 2010)	34
Figura 3.7 Arquitetura de Controle Supervísório (QUEIROZ e CURY, 2002)	38
Figura 3.8 Exemplo da estrutura do código de implementação em Ides2ST	40
Figura 4.1 Metodologia	44
Figura 4.2 Sinótico para a CFM	53
Figura 4.3 SFC Main (VIEIRA et al, 2007)	55
Figura 4.4 Implementação de histórico de eventos	58
Figura 4.5 Passo 1: Criação da fila	59
Figura 4.6 Trecho de código do sistema produto	59
Figura 4.7 Passo 2: Criação de ponteiro para retirar elementos da fila..	60
Figura 4.8 Passo 3: Cálculo da posição que deve ser lida	61
Figura 4.9 Passo 4: Atualização de ponteiro com a nova posição a ser lida	62
Figura 4.10 Passo 5: Geração de histórico de eventos alfanumérico para a CFM	62
Figura 4.11 Trecho de código do evento controlável e_afur	63
Figura 4.12 Código que implementa alarmes gerais	64
Figura 4.13 Gráfico de tendência para a CFM	65
Figura 4.14 Código que implementa a receita para a CFM	66
Figura 4.15 Representação gráfica da receita para a CFM	66
Figura 4.16 Relatório do histórico de eventos para a CFM	67
Figura 4.17 Scripts para implementar número de peças de entrada	68
Figura 5.1 CFM sistema RHINO	72
Figura 5.2 Conjunto final	73
Figura 5.3 Arquitetura sistema RHINO	75
Figura 5.4 Geradores alimentadores (a) G_1 , (b) G_2 , (c) G_3 e (d) G_4	79
Figura 5.5 Geradores do torno e usinagem (a) G_5 e (b) G_7	80
Figura 5.6 Gerador do teste G_6	80
Figura 5.7 Gerador do robô XR4 G_8	81

Figura 5.8 Gerador da esteira de alimentação G_9	81
Figura 5.9 Gerador da aduana G_{10}	82
Figura 5.10 Gerador do robô SCARA G_{11}	82
Figura 5.11 Gerador da esteira de saída G_{12}	83
Figura 5.12 Especificação de coordenação dos alimentadores com o robô XR4 (a) E_{1a} , (b) E_{1b} (c) E_{1c} e (d) E_{1d}	83
Figura 5.13 Especificação do <i>buffer</i> do torno (a) E_{2a} e sistema de visão E_{2b}	84
Figura 5.14 Especificação do <i>buffer</i> do centro de usinagem E_{2c}	84
Figura 5.15 Especificações de coordenação do robô XR4 com o torno (a) E_{3a} e (c) E_{3c} e centro de usinagem (b) E_{3b}	85
Figura 5.16 Especificações de coordenação do torno e centro de usinagem com o robô XR4 (a) E_{4a} , (b) E_{4b} e (c) E_{4c}	85
Figura 5.17 Especificações de coordenação do robô XR4 com o sistema de visão (a) E_{5a} , (b) E_{5b} e (c) E_{5c}	85
Figura 5.18 Especificações de coordenação do sistema de visão com o robô XR4 (a) E_{6a} , (b) E_{6b} , (c) E_{6c} e (d) E_{6d}	86
Figura 5.19 Especificações genéricas para a coordenação do robô XR4 com a esteira de alimentação E_{7a} e com a aduana E_{7b}	87
Figura 5.20 Sequência de operações relacionada à especificação E_{7a}	88
Figura 5.21 Especificação do avanço da esteira E_8	89
Figura 5.22 Especificações do mutex do robô XR4 e SCARA com a esteira de alimentação (a) E_{9a} , (b) E_{9b} e da aduana com a esteira de alimentação (c) E_{9c}	90
Figura 5.23 Especificações de montagem de conjuntos na posição 1 (a) E_{10a} e na posição 2 (b) E_{10b} da mesa de montagem ..	90
Figura 5.24 Especificação da prioridade das bases sobre os anéis e os cilindros E_{11}	91
Figura 5.25 Especificação de limite de duas bases E_{12}	92
Figura 5.26 Especificação esteira de saída E_{13}	93
Figura 5.27 Sinótico para o sistema RHINO	99
Figura 5.28 Tela de comandos estado Manual	101
Figura 5.29 Histórico de eventos gerado para o sistema RHINO	101
Figura 5.30 Trecho de código do evento controlável I_tor_C	102
Figura 5.31 Código que implementa o alarme da esteira de alimentação	103
Figura 5.32 Código que implementa uma receita no sistema RHINO	103

Figura 5.33 Tela da receita	104
Figura 5.34 Trecho de código contagem de anéis aprovados e reprovados	104
Figura 5.35 Tendências de aprovação e reprovação de peças	105
Figura 5.36 Relatório do histórico de eventos do sistema RHINO	105

LISTA DE TABELAS

Tabela 3.1	Eventos da Célula Flexível de Manufatura	30
Tabela 3.2	Plantas locais para a CFM (SILVA, 2010)	35
Tabela 3.3	Número de estados dos geradores da síntese de supervisores (SILVA, 2010)	36
Tabela 4.1	Descrição dos estados relevantes dos supervisores para o sinótico da CFM	52
Tabela 4.2	Código em ST para implementar comandos	56
Tabela 5.1	Eventos do sistema RHINO	76
Tabela 5.2	Plantas locais para o sistema RHINO	94
Tabela 5.3	Número de estados dos geradores da síntese dos supervi- sores	95
Tabela 5.4	Descrição dos estados relevantes de supervisores para o sinótico do sistema RHINO	98

LISTA DE ABREVIATURAS

CFM	célula flexível de manufatura
CLP	controlador lógico programável
e.r.a	em relação a
ERP	<i>enterprise resource planning</i> (planejamento dos recursos da empresa)
IHM	<i>human machine interface</i> (interface humano máquina)
IED	dispositivo eletrônico inteligente
LAN	<i>local area network</i> (rede de área local)
MES	<i>manufacturing execution systems</i> (sistemas de execução da manufatura)
MMI	<i>man machine interface</i> (interface homem máquina)
MTU	unidade terminal mestre
RTU	unidade terminal remota
SCADA	<i>supervisory control and data acquisition</i> (controle supervisorio e aquisição de dados)
SED	sistema a eventos discretos
TCS	teoria de controle supervisório
WAN	<i>wide area network</i> (rede de longa distância)

LISTA DE SÍMBOLOS

\parallel	operador de produto síncrono
\emptyset	conjunto vazio
K, L	linguagens
Σ	alfabeto
Σ_c	conjunto de eventos controláveis
Σ_u	conjunto de eventos não controláveis
Σ^*	conjunto de todas as cadeias finitas de Σ
ε	cadeia vazia
G	gerador
$L(G)$	linguagem gerada por G
$L_m(G)$	linguagem marcada por G
E	especificação
S	supervisor
S_{loc}	supervisor local
S_{red}	supervisor reduzido
S/G	supervisor S controlando G
$SupC(K, G)$	suprema sublinguagem de K controlável e.r.a G

SUMÁRIO

1. INTRODUÇÃO	1
2. SISTEMAS SCADA	7
2.1 DEFINIÇÃO DE SCADA	7
2.2 ARQUITETURA DE UM SISTEMA SCADA.....	8
2.2.1 Operador	9
2.2.2 Interface Homem Máquina (IHM).....	9
2.2.3 Estação central ou unidade terminal mestre (MTU)	9
2.2.4 Rede de comunicação	10
2.2.5 Estação Remota.....	11
2.2.6 Dispositivos de campo	12
2.2.7 Processo físico	12
2.3 FUNÇÕES DO SISTEMA SCADA.....	12
2.4 SOFTWARE DE SUPERVISÃO.....	13
2.4.1 Funcionalidades de um sistema SCADA	14
2.4.2 ScadaBR.....	16
2.5 METODOLOGIAS PARA DESENVOLVIMENTO DE SISTEMAS SCADA	17
2.6 CONCLUSÃO DO CAPÍTULO	18
3. TEORIA DE CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS	21
3.1 SISTEMAS A EVENTOS DISCRETOS (SED)	21
3.2 TEORIA DE CONTROLE SUPERVISÓRIO (TCS)	23
3.3 CÉLULA FLEXÍVEL DE MANUFATURA.....	27
3.3.1 Descrição do sistema.....	28

3.3.2 Modelagem da planta.....	29
3.3.3 Modelagem das especificações.....	32
3.3.4 Síntese de supervisores modulares locais da CFM	34
3.4 ARQUITETURA DE CONTROLE SUPERVISÓRIO	36
3.5 FERRAMENTAS PARA SIMULAÇÃO E IMPLEMENTAÇÃO	
37	
3.5.1 IDES	38
3.5.2 TCT.....	38
3.5.3 Suprema	38
3.5.4 Ides2ST	39
3.6 CONCLUSÃO DO CAPÍTULO	41
4. METODOLOGIA PARA DESENVOLVIMENTO INTEGRADO DE	
SISTEMAS SCADA COM CONTROLE SUPERVISÓRIO	43
4.1 METODOLOGIA	43
4.1.1 Fase I – Projeto informacional.....	45
4.1.2 Fase II – Síntese de controle supervisório	45
4.1.3 Fase III - Emulação.....	47
4.1.4 Fase IV - Implementação do controle supervisório em CLP ..	47
4.1.5 Fase V - Implementação de funcionalidades básicas do sistema	
SCADA 48	
4.1.6 Fase VI - Avaliação de funcionamento do sistema real.....	49
4.1.7 Fase VII - Implementação de funcionalidades gerais do sistema	
SCADA 49	
4.1.8 Fase VIII - Validação.....	50
4.2 IMPLEMENTAÇÃO DE FUNCIONALIDADES DO SISTEMA	
SCADA.....	50
4.2.1 Sinótico.....	50
4.2.2 Envio de Comandos	54
4.2.3 Histórico de Eventos.....	57
4.2.4 Geração de alarmes críticos	63

4.2.5	Geração de alarmes gerais.....	64
4.2.6	Gráficos de tendências	64
4.2.7	Receitas	65
4.2.8	Relatórios	67
4.2.9	Geração de informação para níveis gerenciais.....	67
4.3	CONCLUSÃO DO CAPÍTULO	69
5.	APLICAÇÃO DA METODOLOGIA A UMA CÉLULA FLEXÍVEL DE MANUFATURA	71
5.1	FASE I: PROJETO INFORMACIONAL	71
5.2	FASE II: SÍNTESE DE CONTROLE SUPERVISÓRIO.....	76
5.2.1	Etapa 2.1 Modelagem dos subsistemas da CFM.....	76
5.2.2	Etapa 2.2 Modelagem das especificações	83
5.2.3	Etapa 2.3 Síntese de Supervisores.....	93
5.3	FASE III: EMULAÇÃO.....	96
5.4	FASE IV: IMPLEMENTAÇÃO DO CONTROLE SUPERVISÓRIO EM CLP.....	96
5.5	FASE V: IMPLEMENTAÇÃO DE FUNCIONALIDADES BÁSICAS DO SISTEMA SCADA.....	97
5.6	FASE VI: AVALIAÇÃO DO FUNCIONAMENTO DO SISTEMA REAL	102
5.7	VII: IMPLEMENTAÇÃO DE FUNCIONALIDADES DO SISTEMA SCADA	102
5.8	FASE VIII: VALIDAÇÃO.....	106
5.9	ANÁLISE DE RESULTADOS.....	106
6.	CONCLUSÕES.....	109
	REFERÊNCIAS.....	113

1. INTRODUÇÃO

Com o aumento da competição no mercado, torna-se obrigatório reduzir continuamente os custos e melhorar o processo produtivo sempre que possível. O sucesso depende muito da habilidade para acessar, entender e interpretar o grande volume de informações geradas pela operação do processo. Por isso, surge nas indústrias a necessidade de agregar tecnologia da informação aos processos de automação industrial, abrangendo também a automação do negócio. Em geral a integração entre as diversas tecnologias de informação é organizada em níveis hierárquicos conforme a pirâmide da automação (MORAES e CASTRUCI, 2008), apresentada na Figura 1.1.

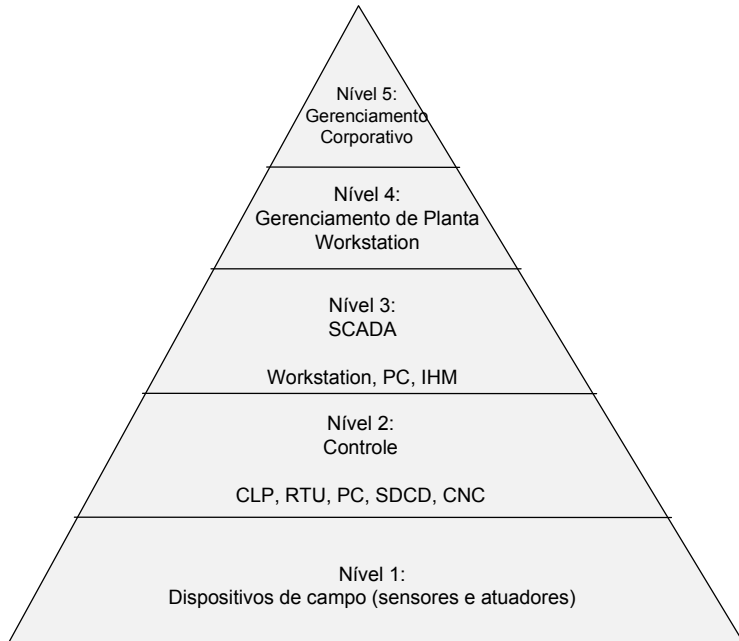


Figura 1.1 Pirâmide da automação

Fonte: Adaptado de Moraes e Castrucci (2008)

Observa-se, na Figura 1.1, os 5 níveis da pirâmide da automação. O nível 1, também chamado “chão de fábrica”, por ser o nível em que estão as máquinas diretamente responsáveis pela produção, é composto

pelos dispositivos de campo: sensores e atuadores. O nível 2 é responsável pelo controle de todos os equipamentos de automação do nível 1 e engloba os Controladores Lógicos Programáveis (CLP), os Sistemas de Controle Distribuído (SDCD), Unidades Terminais Remotas (RTU), e Dispositivos Eletrônicos Inteligentes (IED), entre outros. Esses equipamentos também são responsáveis por repassar os comandos dos níveis superiores para as máquinas da planta da fábrica (nível 1). No nível 3, encontram-se os sistemas supervisórios, comumente conhecidos como sistemas SCADA (*Supervisory Control and Data Acquisition*), os quais concentram as informações passadas pelos equipamentos dos níveis 1 e 2 em banco de dados e as repassam para os níveis administrativos (níveis 4 e 5). O nível 4 é responsável pelo planejamento e programação da planta fabril, passando as tarefas que devem ser realizadas para o nível 3 que, por sua vez, distribui o trabalho para os níveis inferiores. Também é o nível responsável pelo controle e logística de suprimentos. Já a administração de todos os recursos da empresa está no nível 5, no qual é gerenciado todo o sistema. Neste nível encontram-se os *softwares* para gestão de vendas financeiras (MORAES e CASTRUCCI, 2008).

Os sistemas SCADA, podem ser definidos como uma tecnologia que permite que seja monitorada e rastreada a informação de um processo produtivo ou instalação física. Tais informações são coletadas através de equipamentos de aquisição de dados e, em seguida, manipuladas, analisadas, armazenadas e, posteriormente, apresentadas ao usuário. Estes sistemas são tipicamente compostos por estações terminais mestres (MTU), unidades remotas como: CLP, RTU ou IED, dispositivos de campo como: sensores e atuadores e toda a tecnologia de comunicação envolvida para interconexão desses componentes (SILVA e SALVADOR, 2005).

Dentro das funcionalidades dos sistemas SCADA, encontram-se:

- Configuração da interface de comunicação, das variáveis de entrada e de saída e dos operadores que terão acesso ao sistema;
- Construção de sinóticos, que representam graficamente os processos industriais;
- Geração de gráficos de tendências, que acompanham a evolução das variáveis do sistema;
- Geração de alarmes, que avisam ao usuário do sistema quando uma variável ou condição do processo de produção está fora dos valores previstos;
- Geração de históricos, que registram os eventos relevantes;

- Produção de relatórios a partir dos dados da planta, dos alarmes ocorridos em um intervalo de tempo e dos acessos ao sistema por parte dos operadores;
- Geração de *scripts* ou programas, que desenvolvem alguma lógica para o controle do processo;
- Criação de receitas, que configuram os equipamentos da planta para que essa passe a produzir determinado(s) produto(s); e
- Geração de informação para níveis gerenciais.

Estas funcionalidades tornam a maioria dos sistemas SCADA, hoje instalados, uma parte essencial da estrutura de gestão das informações corporativas e do chão de fábrica, os quais são vistos pela gerência, não simplesmente como ferramentas operacionais, mas como recurso importante de informação. O presente trabalho aplica essas funcionalidades para a supervisão e controle de sistemas de manufatura.

Um sistema de manufatura é normalmente composto por um grande número de subsistemas, cuja principal função é desempenhar uma série de atividades, nas quais se desenvolvem operações de montagem e processamento sobre uma matéria-prima, peça ou conjunto de peças de modo a obter-se um produto final (GROOVER, 2001).

É desejável que esses sistemas sejam flexíveis e modulares, de forma a poder absorver as mudanças na demanda, além de serem facilmente aperfeiçoados. No entanto, é natural que isso implique aumento da complexidade nas operações desenvolvidas. A flexibilidade advém do elevado número de tarefas possíveis de serem realizadas e a complexidade é resultado das ações de coordenação dos subsistemas de forma que uma série de tarefas individuais e conjuntas sejam atendidas garantindo o correto funcionamento global e o alcance dos objetivos de produção (BALIEIRO, 2007).

Sob a ótica do problema de coordenação, a dinâmica dos sistemas de manufatura pode ser considerada como dirigida por eventos discretos o que os enquadra na classe de Sistemas a Eventos Discretos (SED). Um SED é um sistema de estados discretos e dirigido a eventos, isto é, a evolução dos seus estados depende, na sua totalidade, da ocorrência de eventos discretos assíncronos ao longo do tempo (CASSANDRAS e LAFORTUNE, 1999). Além dos sistemas de manufatura, os exemplos de aplicações de SED são diversas (RAMADGE e WONHAM, 1989; CASSANDRAS e LAFORTUNE, 1999; CURY, 2001), dentre as quais podem ser ressaltadas: controle de tráfego, telecomunicações, transporte, logística e computação.

Num SED, a função de coordenação é executada pelo sistema de controle que, muitas vezes, é projetado de forma intuitiva ou sustentado pela experiência de profissionais que, em sua grande maioria, não fazem uso de ferramentas que possibilitem uma estruturação do projeto (GOUYON et al., 2004). A não utilização de técnicas baseadas em ferramentas formais para o projeto lógico de controladores conduz a dificuldades no entendimento, manutenção, alteração e detecção de erros no programa de controle, sendo de grande importância aqueles relacionados à segurança operacional do processo (CHANDRA et al., 2003).

Nesse sentido tem ocorrido o desenvolvimento de muitos trabalhos voltados ao estudo e análise de SEDs. Dentre as opções formais existentes para a análise e síntese de controle para SEDs encontram-se: lógica temporal, redes de Petri, max-plus e teoria de controle supervisório. Para avaliar desempenho destes modelos têm-se cadeias de markov, processos semi-markovianos generalizados, redes de filas e simulação (CASSANDRAS e LAFORTUNE, 1999).

A Teoria de Controle Supervisório (TCS) proposta por Ramadge e Wonham (1989), propicia um processo automático de síntese de controladores ótimos para o sistema. Seu objetivo é projetar um supervisor que, em malha fechada com a planta, gere um comportamento que respeite as especificações de controle, garantindo que o sistema funcione conforme o desejado. O sistema final projetado é minimamente restritivo e não bloqueante, ou seja, o supervisor desabilita apenas os eventos da planta que violam as restrições impostas a esta de forma a atribuir um maior grau de liberdade ao sistema controlado.

Dentre as diversas extensões da TCS, está à abordagem modular local (QUEIROZ e CURY, 2002) que possibilita a obtenção de vários supervisores ou elementos de controle menores, os quais oferecem mais clareza na lógica de controle tornando fácil sua interpretação, além de mais confiável ao projetista. Normalmente esses supervisores são menos complexos, uma vez que restringem parcialmente o processo, explorando, portanto, a modularidade da planta e das especificações. Como resultado, novos controladores podem ser rápida e automaticamente projetados, o que facilita alterações no programa do CLP, sempre que é feita alguma modificação (inclusão/exclusão de equipamentos ou alteração no layout) no sistema de manufatura.

Esta abordagem tende a diminuir a complexidade computacional da síntese, além de facilitar a implementação do sistema de controle. Essa implementação é baseada numa Arquitetura de Controle Supervisório (ACS) de três níveis: supervisores modulares, sistema produto e sequências operacionais, proposta por Queiroz e Cury (2002), a qual é

aplicada a uma célula de manufatura real controlada por CLP. Trabalhos recentes têm buscado estender essa estrutura para controle distribuído em múltiplos CLPs (Vieira, 2007). O trabalho de Busseti e Santos (2006) trata da sistematização da implementação gradativa de sistemas automatizados e integrados de manufatura comandados por CLP, de acordo com ACS. Este trabalho mostra um ciclo de desenvolvimento para sistemas re-configuráveis quando novas aplicações forem necessárias, o qual é caracterizado por três etapas: modelagem, síntese e implementação até o atendimento da aplicação demandada para o sistema real.

Como suporte à etapa de implementação deste ciclo de desenvolvimento e com o objetivo de validar com agilidade a ACS implementada num CLP central e vários CLP locais, Diogo et al. (2008) propõem o desenvolvimento de uma aplicação chamada Ambiente Dinâmico (AD), capaz de coletar eventos gerados pela ACS e enviá-los a uma plataforma de simulação e vice-versa. O AD é uma aplicação instalada num computador que é composto por um software SCADA, que desempenha várias funções como fazer a ligação entre o CLP (central e local) e a plataforma de simulação; implementar uma biblioteca de modelos de subsistemas reutilizáveis juntamente com um canal de comunicação para comandos e outro para respostas; promover o encapsulamento dos sinais reais e virtuais numa base de dados para serem usados na simulação. O SCADA possui ainda uma interface com o usuário por cada subsistema, para a escolha entre o modo de Simulação ou modo Ambiente Real onde a simulação deve ocorrer paralelamente à execução do subsistema físico real.

A revisão bibliográfica não aponta estudos que abordem o desenvolvimento integrado de sistemas SCADA que explorem a arquitetura do controle supervisão em CLP.

A integração pode trazer vantagens tanto para o desenvolvimento e aplicação da ACS em CLP quanto para a estruturação de sistemas SCADA. As primeiras estão relacionadas às funcionalidades do sistema SCADA, as quais oferecem ao controle supervisão: visualização do comportamento e modos de operação do sistema por meio de um sinótico; condução do sistema ao estado inicial, supervisionado e manual, através de comandos; geração seletiva de eventos controláveis no estado manual; geração de receitas e; geração de tendências e relatórios dos eventos controláveis e não controláveis para o tratamento de falhas, entre outras. Já as segundas, referem-se, à estruturação da lógica de controle na forma de geradores, a qual permite que informações do sistema real, sejam coletadas a partir dos estados dos modelos das plantas e

supervisores, em forma de sinótico, de alarmes e de histórico de eventos.

Assim, o objetivo geral desta dissertação é desenvolver uma metodologia de implementação de Sistemas SCADA integrada à programação do controle supervísório em CLP para sistemas de manufatura. Como objetivos específicos deste trabalho encontram-se:

- Desenvolver uma implementação preliminar de SCADA aplicada a um problema de controle solucionado por Silva e Queiroz (2009) de uma célula flexível de manufatura (CFM), utilizando a abordagem modular local;
- Propor um método de implementação de funcionalidades de SCADA que fazem proveito da estrutura de implementação de controle supervísório modular local.
- Desenvolver uma metodologia de desenvolvimento de controle supervísório em CLP integrado ao SCADA.
- Aplicar a metodologia proposta ao controle de CFM no laboratório de Automação e Informática Industrial do Departamento de Automação e Sistemas (DAS) da UFSC.

O presente trabalho está dividido em 6 capítulos, sendo o Capítulo 1 a introdução que já foi apresentada. A partir deste ponto, é organizado como segue. No Capítulo 2 apresentam-se a definição, arquitetura, funções, funcionalidades e o software SCADA que implementa as aplicações dos Capítulos 4 e 5, assim como uma revisão bibliográfica de metodologias de implementação de sistemas SCADA em processos produtivos. No Capítulo 3 são descritos conceitos fundamentais da TCS que depois é exemplificada a partir de um problema real de controle solucionado por Silva e Queiroz (2009) para uma CFM; após este exemplo, é explicada a arquitetura de controle supervísório (ACS) são descritas as ferramentas computacionais para realizar a modelagem, síntese e implementação da estrutura genérica de controle em CLP. No Capítulo 4, é proposta a metodologia, objeto deste trabalho, para o desenvolvimento integrado de sistemas de SCADA com controle supervísório, ao tempo que é proposto um método de implementação de funcionalidades de SCADA que fazem proveito da estrutura de implementação de controle supervísório modular local. A aplicação da metodologia a uma CFM é apresentada no Capítulo 5. Finalmente, no Capítulo 6, as conclusões e perspectivas futuras são apresentadas.

2. SISTEMAS SCADA

Os sistemas SCADA são sistemas que utilizam tecnologias de computação e comunicação para automatizar o monitoramento e o controle de processos industriais. Estes sistemas são parte integrante da maioria dos ambientes industriais complexos ou geograficamente dispersos, na medida em que podem coletar rapidamente os dados de uma quantidade grande de fontes, para depois serem apresentados a um operador de uma forma amigável. Os sistemas SCADA melhoram a eficácia do processo de monitoramento e controle, fornecendo a informação oportuna para poder tomar decisões operacionais apropriadas (PINHEIRO, 2006).

Neste capítulo são apresentadas as principais características e funcionalidades de um sistema SCADA, além de serem descritos os requisitos comuns a estes sistemas, tanto de software quanto de hardware. Finalmente, são apresentadas metodologias para desenvolvimento e implementação de sistemas SCADA.

2.1 DEFINIÇÃO DE SCADA

O termo SCADA vem do inglês “Supervisory Control And Data Acquisition”, ou Controle Supervisório e Aquisição de Dados e se define como um sistema que permite supervisionar e controlar um processo produtivo ou instalação física, através da troca de informação entre uma estação central (chamada também de Unidade Terminal Mestre, MTU), e uma ou mais unidades remotas (CLP, RTU, IED). Através desses equipamentos é feita a aquisição de dados dos dispositivos de campo, fazendo-se necessário a utilização de redes de comunicação para atingir este objetivo. As informações coletadas são manipuladas, analisadas, armazenadas e, posteriormente apresentadas ao operador em multiplicidade de formas, através de uma interface de alto nível (COELHO, 2010).

2.2 ARQUITETURA DE UM SISTEMA SCADA

A arquitetura típica de um sistema SCADA está relacionada com seus componentes (KRUTZ, 2006), os quais são esquematicamente apresentados na Figura 2.1, os quais são explicados a seguir.

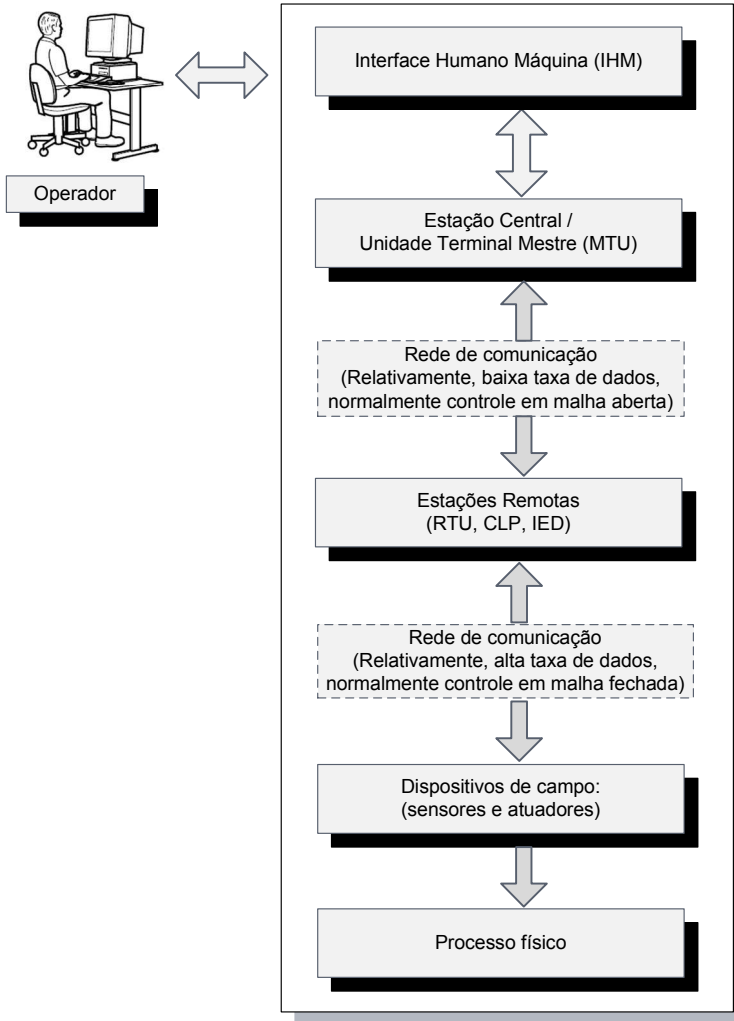


Figura 2.1 Arquitetura típica de um sistema SCADA

Fonte: Adaptado de Krutz (2006)

2.2.1 Operador

O operador humano monitora e interage com o sistema SCADA e executa remotamente as funções de controle supervisiório.

2.2.2 Interface Homem Máquina (IHM)

A interação entre os operadores e a estação central, é efetuada através de uma Interface Homem-Máquina (IHM). Esta interface é constituída por software e hardware, que permite aos operadores monitorar o estado de um processo, modificar os valores de referência (*setpoint*), e suspender manualmente as operações de controle automático em casos de emergência.

A IHM, apresenta graficamente as informações do processo na forma de sinóticos onde o operador pode visualizar um diagrama esquemático da planta que é controlada, a representação gráfica das estações remotas, os valores atuais dos instrumentos fabris e a apresentação dos alarmes ativos.

Dado que é ligada com o banco de dados do sistema SCADA, a IHM promove registros, diagnóstico de dados e informação de administração como: procedimentos de manutenção, informação de logística, detalhes de agendamento e guias para resoluções de problemas.

2.2.3 Estação central ou unidade terminal mestre (MTU)

A estação central ou MTU é a unidade principal do sistema SCADA sendo equivalente a uma unidade mestre numa arquitetura mestre-escravo e responsável principalmente por coletar, armazenar e processar as informações geradas pelas estações remotas, para depois serem colocadas à disposição dos diversos clientes ou operadores que possam requerê-las através da IHM e dessa forma agir em conformidade com os eventos detectados. A taxa de transmissão de dados entre a MTU e a estação remota é relativamente baixa e o controle é usualmente em malha aberta devido a possíveis atrasos de comunicação ou interrupção no fluxo dos dados (KRUTZ, 2006).

A MTU pode estar centralizada em um único computador (servidor SCADA) ou distribuída em uma rede de computadores ou clientes, de modo a permitir a partilha de informações proveniente do servidor SCADA (BOARETTO, 2008). A MTU pode conter também, dispositi-

vos auxiliares como impressoras e registradores e um ou mais servidores redundantes, para os casos de falha.

A MTU agrupa o conjunto de dados (entrada e saída (I/O), históricos, alarmes, entre outros) e informações em um Banco de Dados (BD). Normalmente este BD está contido no servidor (SCADA) ou pode pertencer a vários servidores conectados em rede. Na maioria das aplicações, é requerido que a MTU envie informações de contabilidade ou de gestão para outros computadores ou sistemas financeiros dentro da empresa.

As conexões entre a MTU e outros sistemas podem ser feitas através de links de comunicação dedicados ou redes de área local (LAN). A Figura 2.2, mostra os componentes e a configuração de um sistema SCADA distribuído, o qual está constituído de uma estação central composta por um servidor SCADA, IHMs e BD entre outros componentes, conectados por uma rede LAN; diversos tipos de rede de comunicação (cabos, sem fio, linhas telefônicas dedicadas, etc) e várias estações remotas (CLP, RTU, IED), os quais são explicados nas próximas seções.

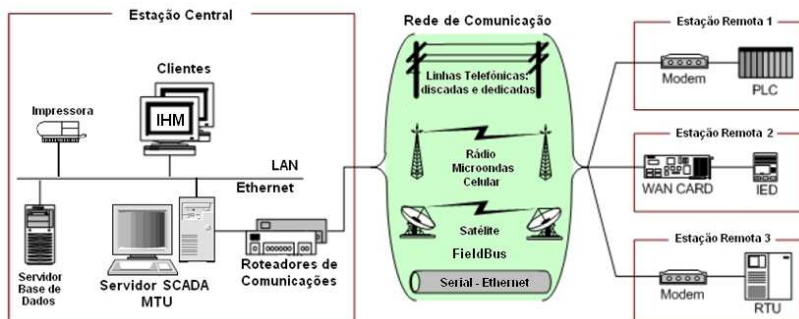


Figura 2.2; **Error! No hay texto con el estilo especificado en el documento.** Layout geral de um sistema SCADA

Fonte: Adaptado de Stouffer, Falco e Kent (2006)

2.2.4 Rede de comunicação

É a plataforma através da qual as informações de um sistema SCADA são transferidas. A fim de comunicar a MTU com as estações remotas, que estão localizadas a distância, ou com os diferentes computadores e sistemas que estão localizados dentro da rede corporativa, um link de comunicação deve existir para transferir dados de um local para

outro. Considerando os requisitos do sistema e as distâncias a cobrir, as redes de comunicação podem ser implementadas através dos seguintes meios físicos: Internet, cabos elétricos ou cabos de fibra óptica (serial, ethernet), redes sem fio, linhas telefônicas dedicadas e discadas, micro-ondas, rádio, satélite, entre outros (PINHEIRO, 2006).

Em geral, a troca de dados é estabelecida através de protocolos de comunicação padrão como o MODBUS nas versões RTU (Padrão serial RS232/RS485) e TCP (Padrão Ethernet), HDLC (High level Data Link Control), DNP3 e IEC 60870. É comum existir uma interface de comunicação padrão chamada OPC (OLE for Process Control) usada no setor de controle de processo, que garante a interoperabilidade entre equipamentos de diferentes fabricantes a qual é baseada em padrões Microsoft OLE, DCOM e RPC. O tipo de esquema de comunicação irá determinar a confiabilidade e desempenho do sistema SCADA (PENIN, 2007).

2.2.5 Estação Remota

Uma estação remota é um conjunto de elementos afastados da MTU e dedicados a tarefas de controle e/ou leitura dos valores atuais dos dispositivos a quem estão associados. Nesta classificação, podem encontrar-se vários elementos mais ou menos diferenciados:

- RTU: especializados em comunicação.
- CLP: especializados em tarefas gerais de controle.
- IED: especializados em tarefas específicas de controle.

A RTU, também chamada de unidade terminal remota, é um dispositivo eletrônico frequentemente equipado com interface de rádio sem fio, que tem a responsabilidade de coletar dados de campo através de suas portas de entrada e saída analógica e digital, que são conectadas aos sensores e atuadores. A RTU também pode controlar equipamentos, como por exemplo, abrindo ou fechando uma chave ou uma válvula, ou definindo a velocidade de uma bomba. A quantidade de dados obtidos pela RTU dependerá do tamanho do sistema a ser controlado e serão enviados para a MTU utilizando o sistema de comunicação. A RTU é especialmente indicada para situações adversas onde a comunicação é difícil (BAILEY e WRIGHT, 2003).

Os CLPs são dispositivos digitais que permitem controlar o processo fabril mediante uma memória programável que reúne as instru-

ções que devem ser repassadas para as máquinas e os equipamentos responsáveis pela produção industrial. A grande vantagem dos CLPs sobre as RTUs é que os primeiros podem ser utilizados em tarefas simples e podem ser facilmente configurados para efetuar várias funções, além de serem compactos, ocupando pouco espaço (BAILEY e WRIGHT, 2003). Atualmente, nota-se uma convergência no sentido de reunir as melhores características desses dois equipamentos: a facilidade de programação e controle dos CLPs e as capacidades de comunicação dos RTUs (BOARETTO, 2008).

Os IEDs são dispositivos eletrônicos que possuem algum tipo de inteligência local. No entanto, especificamente sobre a indústria de automação de sistemas de potência e proteção, o termo realmente veio a existir para descrever um dispositivo versátil que tem funções de proteção elétrica, controle avançado de inteligência local, habilidades de monitoramento e capacidade de comunicação extensiva diretamente com um sistema SCADA (STOUFFER, FALCO E KENT, 2006).

2.2.6 Dispositivos de campo

Os dispositivos de campo referem-se aos sensores e atuadores que estão diretamente interligados à planta ou equipamento a ser controlado e monitorado pelo sistema SCADA. Os primeiros convertem parâmetros físicos, tais como velocidade, níveis de água e temperatura, para sinais analógicos e digitais legíveis pela estação remota, e os segundos são usados para atuar sobre o sistema, ligando e desligando determinados equipamentos.

2.2.7 Processo físico

O processo físico é tanto o elemento que se deseja monitorar e/ou controlar quanto o objeto da automação, onde informações deste elemento são capturadas através de instrumentos, tanto para o controle do processo quanto para a gerência de dados.

2.3 FUNÇÕES DO SISTEMA SCADA

Atualmente os principais sistemas de supervisão oferecem três funções básicas (VIANNA, 2008):

- 1. Funções de supervisão:** Inclui todas as funções de monitoramento do processo tais como: sinóticos animados, gráficos de tendência de variáveis analógicas e digitais, relatórios em vídeo e impressos, etc.
- 2. Funções de operação:** Inclui a ação direta sobre os atuadores permitindo enviar comandos como ligar e desligar equipamentos e sequência de equipamentos, operação de malhas com controle PID, mudança de modo de operação de equipamentos, etc.
- 3. Funções de controle:** Alguns sistemas possuem opções específicas para atuação automática sobre o sistema em determinadas situações pré-programadas de acordo com a necessidade e possibilidade de ter esse tipo de automatismo sobre o processo supervisionado.

2.4 SOFTWARE DE SUPERVISÃO

Um software de supervisão é um pacote de software constituído de um ambiente de desenvolvimento e um programa de execução. O ambiente de desenvolvimento inclui utilidades relacionadas com a criação e edição de janelas de aplicativos diversos e suas características (textos, desenhos, cores, propriedades de objetos, programas, etc). Já o programa de execução ou *run-time* permite executar a aplicação criada com o programa de desenvolvimento (na indústria é entregue como produto final o *run-time* e a aplicação). Esse pacote também inclui os controladores ou *drivers* que permitem a comunicação do software SCADA com os dispositivos de controle da planta e com a rede de gestão da empresa (PENIN, 2007).

O software SCADA identifica os *tags*, que são variáveis numéricas ou alfanuméricas envolvidas na aplicação, podendo executar funções computacionais (operações matemáticas, lógicas, com vetores ou strings, etc) ou representar pontos de entrada/saída de dados do processo que está sendo controlado. Neste caso, correspondem às variáveis do processo real (ex: temperatura, nível, vazão etc), se comportando como a ligação entre o controlador e o sistema. É com base nos valores das *tags* que os dados coletados são apresentados ao usuário (SILVA e SALVADOR, 2005).

O software de supervisão pode ser proprietário ou livre. Geralmente os proprietários são desenvolvidos para se comunicar com os equipamentos desenvolvidos por eles próprios e por isso precisa-se adquirir a solução completa de *hardware* e *software* de um só fornecedor.

Por outro lado existem os de código aberto que utilizam os protocolos padrões de comunicação com as RTUs e ou CLPs possibilitando a utilização de RTUs fornecidas por fabricantes diferentes proporcionando uma liberdade maior na escolha de novas RTUs ou CLPs obedecendo a critérios de funcionalidade e preço (ALMEIDA, 2009).

2.4.1 Funcionalidades de um sistema SCADA

Os sistemas SCADA possuem um ambiente integrado de desenvolvimento que possui editor de gráficos, editor para banco de dados, relatórios, receitas e editor de *scripts*. A seguir são descritas as funcionalidades mais implementadas em um software SCADA:

- **Configuração**
Permite definir o ambiente de trabalho para atender às necessidades da aplicação. São configurados tanto o protocolo de comunicação, quanto os pontos de I/O ou *tags*. A estrutura de telas é organizada da forma mais conveniente. Os usuários são classificados de acordo com sua importância, com privilégios que definem a sua influência sobre o sistema.
- **Sinótico ou interface gráfica**
Os sinóticos permitem a elaboração de telas de usuário com múltiplas combinações de imagens e/ou textos, definindo assim as funções de controle e supervisão da planta. Cada sinótico representa uma área do processo em certo nível de detalhe, incluindo o valor em tempo real das variáveis presentes na planta.
- **Gráficos de Tendências**
São gráficos X-Y dos valores das *tags* armazenadas que permitem representar de forma fácil a evolução de variáveis do sistema, contínuas ou discretas. É possível, em um gráfico, representar vários valores de forma simultânea. Os períodos de amostragem que variam tipicamente de 100 ms a 1 hora devem ser escolhidos de acordo com a velocidade real do processo. Quando se deseja armazenar valores de variáveis em disco por longos períodos de tempo (até 1 ano) se recorre ao registro histórico.

- **Alarmes**

Este módulo está presente em todos os sistemas SCADA. Ele recebe os eventos excepcionais do processo e os registra identificando: data e hora do evento, variável alarmada, valor no momento do alarme, descrição do evento, data e hora de normalização do evento e status do evento (alarmado, normalizado). Uma janela de alarmes exibe os alarmes mais recentes. Quando um alarme ocorre o operador é avisado através de uma buzina, música ou por um *speech maker*. O operador deve declarar que está ciente do problema, reconhecendo o alarme mais recente ou todos os alarmes simultaneamente.

- **Históricos**

Registram em arquivos todos os eventos relevantes de operação, com data, hora, descrição do evento e operador “logado” na hora do evento, assim como também as ocorrências de alarmes, gráficos de tendências, etc. Os eventos de interesse geralmente são de configuração da base de dados, de operação críticos (ação sobre malhas de controle, partida e parada da planta) e de equipamentos críticos, etc.

- **Relatórios**

Cada vez é mais comum a tendência a complementar as funcionalidades de aquisição, registro de dados e geração de alarmes com a capacidade de gerar informação, em forma de relatórios, capaz de ajudar na tomada de decisões. Por exemplo, pode ser interessante dispor de informação referente à situação da planta (estado, incidências), produção em tempo real, geração e registro de alarmes, aquisição de dados para análises históricas, de controle de qualidade, cálculo de custos, manutenção preventiva, etc. O usuário deve definir as variáveis que farão parte do relatório e o seu período (ou instante) de amostragem.

- **Controle de processo através de scripts**

Linguagens de programação de alto nível, como Visual Basic, C ou Java, incorporados nos software SCADA, permitem programar tarefas que respondam a eventos do sistema, como enviar comandos ao sistema de controle para ligar ou desligar equipamentos em função dos valores das variáveis adquiridas, enviar um e-mail ao ativar-se um alarme concreto ou simplesmente

iniciar rotinas no dispositivo de controle (CLP, RTU, etc), associadas, por exemplo, ao início do sistema de controle e modos de funcionamento.

- **Receitas**

Trata-se de arquivos que salvam os dados de configuração dos diferentes elementos do sistema (velocidade do processo, pressões, temperatura, níveis de alarme, quantidade de peças produzidas, etc.). Dessa forma, o procedimento de mudar a configuração de trabalho de toda uma planta ficará reduzido ao simples fato de clicar um botão após confirmar os dados de acesso (usuário, senha e número ou nome de receita, por exemplo). O sistema SCADA será responsável por enviar os dados aos correspondentes controladores, ficando a planta pronta para as novas condições de trabalho.

- **Geração de informação para níveis gerenciais**

Esta funcionalidade permite gerar informações como volumes e taxas de produção, número de produtos com defeito, entre outras, que auxiliem na tomada de decisões em níveis mais altos da pirâmide de automação, como sistemas MES e ERP.

2.4.2 ScadaBR

Conforme mencionado, os softwares SCADA estão disponíveis em pacotes comerciais ou proprietários com um custo muito elevado, o que muitas vezes inviabiliza sua utilização para automação de sistemas. Porém, no final da década de 90 surgiram as primeiras iniciativas em software livre com algumas funcionalidades de SCADA, e recentemente o movimento tomou mais força, especialmente nos últimos 5 anos com tecnologias mais recentes e projetos de maior porte (ROCHA, 2011). Neste trabalho adota-se um dos softwares de código aberto disponíveis, o ScadaBR, para objetivar a aplicação da metodologia proposta.

O software ScadaBR foi desenvolvido em modelo "open-source", pela Fundação Centros de Referência em Tecnologias Inovadoras (CERTI), possuindo licença gratuita. Toda a documentação e o código-fonte do sistema estão à disposição (<http://www.scadabr.com.br/>), inclusive sendo permitido modificar e re-distribuir o software se necessário.

O ScadaBR é uma aplicação multiplataforma baseada em Java, ou seja, é possível executá-la usando o Windows, Linux ou outros sis-

temas operacionais, a partir de um servidor de aplicações (sendo o Apache Tomcat a escolha padrão). Ao executar o aplicativo, ele pode ser acessado a partir de um navegador de Internet, preferencialmente o Firefox ou o Chrome. A interface principal do ScadaBR é de fácil utilização e já oferece visualização das variáveis, gráficos, estatísticas, configuração dos protocolos, alarmes, construção de telas tipo IHM e uma série de opções de configuração. Este software oferece comunicação em mais de 20 protocolos incluindo: OPC, Modbus, ASCII, DNP3, IEC101, Bacnet.

2.5 METODOLOGIAS PARA DESENVOLVIMENTO DE SISTEMAS SCADA

Quando uma empresa decide desenvolver e implementar um sistema SCADA é necessário seguir uma metodologia que permita uma integração de todas as áreas envolvidas no processo produtivo, da melhor forma possível. Observam-se na literatura acadêmica, metodologias onde somente é concebido o desenvolvimento da IHM, mas existem outras, que abrangem desde o planejamento da arquitetura do SCADA até a realização de testes finais de funcionamento, as quais serão apresentadas de forma sucinta, na sequência.

Segundo Moraes e Castrucci (2001) as etapas que devem compor o planejamento de um Sistema Supervisório são: entendimento do processo a ser automatizado; tomada de dados (variáveis); planejamento do banco de dados; planejamento dos alarmes; planejamento da hierarquia de navegação entre telas da IHM; desenho de telas; gráfico de tendências dentro das telas; planejamento de um sistema de segurança e escolha de uma plataforma como Windows ou Linux. Esta metodologia não enfatiza o planejamento de hardware e software, assim como os testes operacionais do sistema.

Por outro lado, Cábus, *et al.* (2004) define cinco fases de implementação: (i) design da arquitetura do sistema (MTU, dispositivos de controle, de campo, tipo de rede, distâncias, número de E/S, protocolos, drivers, etc.); (ii) definição das RTUs necessárias, comunicações, dispositivos IHM e hardware em geral, assim como também a aquisição de um pacote de software SCADA apropriado à arquitetura dos sistemas da planta; (iii) instalação do equipamento de comunicação e MTU; (iv) programação, tanto dos equipamentos de comunicações, quanto dos equipamentos IHM e software SCADA; e por último (v) teste do siste-

ma como um todo, durante o qual os problemas de programação de comunicações com o software SCADA são solucionados.

De acordo com Reyes (2007), o desenvolvimento e implementação de um sistema SCADA deve ser realizado em três fases: definição, especificação e implementação do sistema. Cada fase tem duas etapas. A primeira fase, determina as necessidades e objetivos do processo e posteriormente, define os requerimentos do sistema a implementar. Já na segunda fase, são elaboradas as especificações funcionais para depois selecionar a arquitetura do sistema e seus componentes. Por fim, na terceira fase, é realizada a configuração final do sistema e a instalação é colocada em operação.

Finalmente, uma metodologia melhor estruturada, é a proposta pela (IEEE, 2008), cujo desenvolvimento do sistema requer várias etapas e provavelmente irá exigir várias iterações por parte do projetista. Inicialmente, deve definir-se os requerimentos funcionais do sistema a curto e longo prazo como: medição e monitoração do estado do processo, controle, serviços auxiliares, sincronização de tempo e a lógica a ser programada. O maior benefício dessa definição será obtido através de uma análise a partir do nível empresarial de tal forma que os padrões corporativos possam ser desenvolvidos. Esta definição deve ser baseada na infra-estrutura existente ou prevista e na percepção das necessidades atuais e antecipação das necessidades futuras.

Posteriormente, deve realizar-se a seleção de dispositivos de controle como CLPs, IEDs, RTUs, a seleção da IHM (software e hardware) e o desenvolvimento de telas (sinóticos, alarmes, tendências e histórico de alarmes), a definição de requerimentos de segurança, a seleção da arquitetura e protocolos de comunicação (interna e externa ao processo) bem como definição de disponibilidade de requerimentos para funções do sistema. Por outro lado, a seleção de componentes e arquiteturas é geralmente limitada por vários elementos, tais como equipamentos pré-existentes, normas de procedimento, custos, etc. Portanto, o projeto do sistema pode redefinir requisitos anteriormente estabelecidos, resultando em um processo iterativo. Por último, são realizados testes do sistema em geral, para verificar e avaliar o seu funcionamento a fim de chegar a um compromisso satisfatório.

2.6 CONCLUSÃO DO CAPÍTULO

Um sistema SCADA permite monitorar e rastrear informações de um processo produtivo ou instalação física. Essas informações são cole-

tadas através de equipamentos de aquisição de dados e meios de comunicação e, em seguida, são manipuladas, analisadas, armazenadas e, posteriormente apresentadas ao usuário. Também, o sistema SCADA, pode exercer controle através de ações, que podem efetuar-se remotamente sobre os equipamentos controlados pelo sistema, como por exemplo, abrir e fechar uma válvula, ligar ou desligar um motor, entre outras.

Neste capítulo foram expostas as características e funções principais de um sistema SCADA, assim como o hardware, software, sistema de comunicação e funcionalidades associadas a este. Por último, foram apresentadas algumas metodologias referentes ao desenvolvimento e implementação de sistemas SCADA. Cabe mencionar que, na literatura acadêmica, não foi encontrada uma metodologia para desenvolver sistemas SCADA para sistemas a eventos discretos (SED), que estivesse baseada numa arquitetura de controle supervisão programada num equipamento de controle como o CLP.

3. TEORIA DE CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS

Neste capítulo, inicialmente são explicados os conceitos fundamentais da TCS (RAMADGE e WONHAM, 1989) e da abordagem modular local (QUEIROZ e CURY, 2002). Em seguida, para exemplificar a teoria, é apresentada de forma resumida, a solução de um problema real de controle supervísório de uma célula flexível de manufatura (CFM) (SILVA e QUEIROZ, 2009) que será aproveitada para realizar a implementação preliminar da metodologia proposta no capítulo 4. Posteriormente, é descrita uma arquitetura de controle supervísório (QUEIROZ e CURY, 2000) utilizada para a implementação do sistema de controle em CLP, na qual são preservadas as características originais da TCS e cuja estrutura de controle está dividida em três níveis: Supervisores Modulares, Sistema Produto e Sequências Operacionais. Finalmente, são apresentadas as ferramentas computacionais que auxiliam o processo de síntese de supervisores.

3.1 SISTEMAS A EVENTOS DISCRETOS (SED)

Um Sistema a Eventos Discretos (SED) é um sistema dinâmico que evolui de acordo com a ocorrência abrupta, possivelmente em intervalos irregulares imprevisíveis, de eventos físicos (RAMADGE e WONHAM, 1989).

Como a dinâmica dos SEDs é regida pela ocorrência de eventos, é conveniente que seu comportamento seja representado por linguagens. As linguagens são conjuntos de cadeias, ou palavras, que são sequências de eventos. Define-se que uma linguagem L definida sobre um alfabeto Σ , é um conjunto de cadeias ou palavras formadas por símbolos pertencentes a esse alfabeto Σ . Define-se ainda Σ^* como o conjunto de todas as cadeias finitas de elementos do conjunto Σ possíveis de ocorrer, incluindo a cadeia vazia ε (CURY, 2001).

No contexto das linguagens, um SED pode ser modelado através de geradores. Um gerador é definido como uma quintupla $G = (\Sigma, Q, \delta, q_0, Q_m)$, sendo que os elementos dessa estrutura, respectivamente, são: alfabeto de eventos; conjunto de estados; $\delta: \Sigma \times Q \rightarrow Q$ a função de transição, é uma função parcial definida em cada estado de Q para

um subconjunto de Σ ; $q_0 \in Q$, estado inicial e $Q_m \subseteq Q$, conjunto de estados marcados. Um gerador está associado a duas linguagens: a gerada $L(G)$ que representa o comportamento livre da planta, ou todas as cadeias que podem ser seguidas no gerador, partindo-se do estado inicial, e a marcada $L_m(G)$, que representa as sequências que a partir do estado inicial chegam a um estado marcado, o que significa conclusão de uma tarefa.

Os geradores podem ser graficamente visualizados por diagramas de transição de estado, que são grafos direcionados. Os nós representam os estados, por exemplo, os estados possíveis de uma máquina. E os arcos desse grafo são identificados com os eventos responsáveis pela transição de estados. Um exemplo de grafo pode ser visto na Figura 3.1.

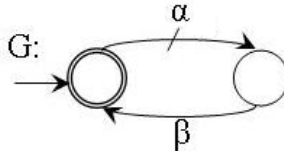


Figura 3.1 Visualização gráfica de um gerador

O gerador da Figura 3.1 pode significar o funcionamento de uma máquina ou o de uma planta, sendo que o estado inicial, representado por um círculo de borda dupla e com uma flecha apontando a ele, designa máquina em repouso, e o estado, representado por um círculo simples, máquina em operação. A transição rotulada com o evento α dita o início de funcionamento da máquina, enquanto que a transição β , o fim de operação.

Um estado de G é acessível se pode ser alcançado através de alguma sequência de eventos tendo como origem o estado inicial q_0 , assim G é acessível se todos os seus estados são acessíveis; por outro lado, um estado de G é co-acessível ou não bloqueante se sempre for possível a partir deste, alcançar um estado marcado com a ocorrência de eventos pertencentes ao alfabeto Σ , portanto G é co-acessível se todos os seus estados são co-acessíveis. Um gerador acessível e co-acessível é dito *Trim*. Ele representa a ausência de bloqueios no sistema, isto é, a partir do estado inicial q_0 ou de qualquer outro estado, sempre existirá um caminho que conduz a um estado marcado.

O bloqueio se dá quando um gerador não consegue avançar até um estado marcado, ou porque mesmo progrediu até um estado não marcado onde não há transições de saída, *deadlock*, ou porque a evolução do sistema fica confinada a um conjunto de estados não marcados a

partir dos quais não existe uma sequência de eventos que leve a um estado marcado, *livelock*.

Segundo a TCS, um SED pode ser visto como a composição de subsistemas sendo regidos por um controlador que restringe o comportamento do sistema de acordo com as especificações impostas no projeto. Assim, a composição síncrona de dois ou mais geradores (CAS-SANDRAS e LAFORTUNE, 1999) é uma operação que permite representar a evolução paralela dos mesmos, com todas as sequências possíveis e tarefas que podem ser completadas pelo sistema. Esta operação sincroniza os eventos compartilhados, enquanto que os não compartilhados pelos geradores podem executar a qualquer momento. A composição síncrona entre dois geradores G_1 e G_2 é simbolizada por $G_1 \parallel G_2$ sendo que é uma operação comutativa ($G_1 \parallel G_2 = G_2 \parallel G_1$) e associativa ($(G_1 \parallel G_2) \parallel G_3 = G_1 \parallel (G_2 \parallel G_3)$) e quando se precisa compor vários geradores a operação pode ser feita repetidas vezes aos pares.

3.2 TEORIA DE CONTROLE SUPERVISÓRIO (TCS)

A TCS proposta por Ramadge e Wonham (1989) propõe um desenvolvimento de modelos formais de sistemas de controle baseado na teoria de autômatos e linguagens, a qual é dotada de procedimentos de síntese automática de supervisores que atendem de forma menos restritiva possível as especificações de controle. A TCS faz uma distinção clara entre o sistema a ser controlado, denominado planta, e a entidade que o controla. A planta é representada por um gerador que reflete o comportamento fisicamente possível dos subsistemas que a compõem. O papel do supervisor 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.

Dentro desta abordagem, considera-se que o supervisor S , representado por um gerador e cujos eventos estão contidos no alfabeto de G , interage com este (planta), numa estrutura em malha fechada, onde S observa os eventos ocorridos e define de acordo com o estado atual da planta, quais eventos fisicamente possíveis são habilitados. Sob esse aspecto, a forma de controle é dita permissiva, no sentido de que eventos inibidos não podem ocorrer, e, os autorizados, não precisam obrigatoriamente ocorrer. O conjunto de eventos habilitados num dado instante pelo supervisor define uma entrada de controle, que é atualizada a cada nova ocorrência de evento observada em G . A Figura 3.2 ilustra a inte-

ração do supervisor com a planta que ocorre através da estrutura em malha fechada.

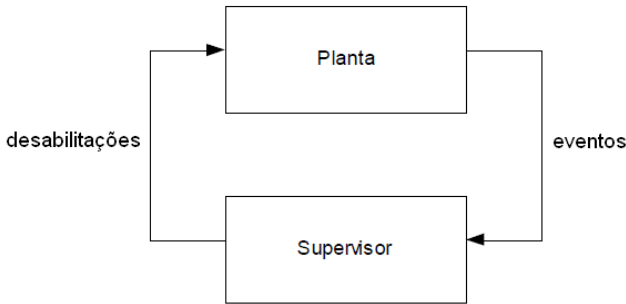


Figura 3.2 Estrutura em malha fechada do controle supervísório

De modo a modelar um sistema de controle, Ramadge e Wonham (1989) particionam o conjunto de eventos Σ em dois subconjuntos, dos eventos controláveis (Σ_c) e dos eventos não controláveis (Σ_u), de modo que $\Sigma = \Sigma_c \cup \Sigma_u$. Os eventos em Σ_c podem ser desabilitados ou habilitados em qualquer momento pelo supervisor S, ao passo que os eventos em Σ_u não são afetados pela ação do supervisor. Os eventos controláveis são representados por um traço no arco de transição. Quando estes eventos são desabilitados, é porque são omitidos nas respectivas transições de estado de S, quando habilitados é devido a que estão explícitos nas transições de estado do supervisor S.

O comportamento do sistema composto pela planta G, sujeita à ação de um supervisor S, é denotada por (S / G) de acordo com Ramadge e Wonham (1989). Um supervisor S é dito ser não bloqueante para um gerador G se $\overline{L_m(S / G)} = L(S / G)$, sendo \overline{L} o conjunto de todas as cadeias de Σ^* que são prefixos de cadeias de L (prefixo-fechamento). Isto implica que, as palavras ou sequências de eventos geradas pelo sistema sob supervisão sempre podem evoluir no sentido de concluir-se uma tarefa imposta pelo sistema.

De um modo geral, um problema de síntese de supervisores supõe que se represente o comportamento fisicamente possível e as restrições impostas ao sistema, sendo o objetivo construir um supervisor não bloqueante para a planta de forma que o comportamento do sistema em malha fechada se limite ao comportamento desejado. Para a realização do supervisor e tendo como base o comportamento em malha aberta da planta G, é obtida uma especificação K, a partir da composição síncrona de G e as especificações genéricas E.

A condição necessária e suficiente para a existência de um supervisor não bloqueante S que satisfaça uma dada especificação $K = L_m(S / G) \subseteq L_m(G)$ é a controlabilidade de K (RAMADGE e WONHAM, 1989). K é controlável (e.r.a G) se $\bar{K}\Sigma_u \cap L(G) \subseteq \bar{K}$, ou seja, a ocorrência de um evento não controlável em G não gera uma sequência de eventos que desrespeite a especificação. Seja $C(K,G)$ o conjunto de todas as sublinguagens de K controláveis e.r.a G . Existe um elemento supremo, chamado $\text{SupC}(K,G)$, que contém a máxima sublinguagem de K que é controlável e.r.a G (WONHAM, 2008). Assim, quando K não é inteiramente controlável, $\text{SupC}(K,G)$ representa o comportamento menos restritivo possível.

Nos problemas de controle mais complexos os SED são modelados por múltiplos subsistemas que devem ser coordenados para atender a um conjunto de especificações. O controle é monolítico quando há um único supervisor atuando sobre a planta que atenda todas as especificações. Uma desvantagem no uso desta abordagem é o crescimento exponencial do número de estados do gerador obtido pela composição síncrona de um grande número de subsistemas que compõe a planta G com um grande número de especificações, requerendo um esforço computacional muito grande para tratar do problema, além de ser difícil de ser implementado no CLP.

Como alternativa, foi desenvolvida uma extensão à abordagem monolítica, chamada de Controle Modular Local (Queiroz e Cury, 2000), a qual permite explorar a modularidade das especificações e da planta, de forma a diminuir a complexidade computacional da síntese de supervisores e o tamanho das soluções. Nesta abordagem, além de decompor o sistema físico em subsistemas modelados por geradores que representam o seu comportamento, existe um diferencial no que diz respeito a que cada especificação imposta para o sistema tem uma planta local, que consiste na composição síncrona dos geradores que compartilham ao menos um evento com esta especificação. Posteriormente, são obtidas as especificações locais, que consiste na composição síncrona da planta local com a especificação que deu origem a esta. Por fim, a ação de controle é dividida em vários supervisores chamados de supervisores locais, os quais são obtidos pelo cálculo da máxima linguagem controlável para cada planta local. Assim cada supervisor coordena uma parte do sistema global.

Conforme Queiroz (2000) e Queiroz e Cury (2000a, 2000b), cada subsistema é representado por um gerador da estrutura $G_i = (\Sigma_i, Q_i, \delta_i, q_{0i}, Q_{mi})$, $i \in I = \{1, \dots, n\}$, sendo n o número de subsiste-

mas. Cada especificação genérica é representada por um gerador E_j , cuja linguagem está contida em Σ_j com $\Sigma_j \subseteq \Sigma$ para $j \in J = \{1, \dots, m\}$ sendo m o número de especificações. Assim, cada planta local $G_{locj} = (\Sigma_{locj}, Q_{locx}, \delta_{locj}, q_{0locj}, Q_{mlocj})$, para $j = 1, \dots, m$, referente a especificação E_j , é representada por $G_{locj} = \parallel_{i \in I_{locj}} G_i$ para, $I_{locj} = \{k \in K \mid \Sigma_k \cap \Sigma_j \neq \emptyset\}$.

Na Figura 3.3 é mostrada uma ilustração da metodologia usada para a obtenção dos supervisores locais. Sejam G_0, G_1, G_2 e G_3 os geradores que representam os subsistemas da planta e E_0, E_1 e E_2 as especificações genéricas de um determinado sistema. Cada especificação genérica está associada a um conjunto I_{locj} formado pelos índices dos subsistemas que compartilham ao menos um evento com esta especificação. Desta forma, E_0 está associada ao conjunto $I_{locE_0} = \{G_0, G_1\}$, E_1 ao conjunto $I_{locE_1} = \{G_1, G_2\}$ e E_2 ao conjunto $I_{locE_2} = \{G_2, G_3\}$. Assim, as plantas locais referentes a cada especificação são obtidas realizando o produto síncrono dos elementos de cada conjunto. Desta forma, o comportamento desejado pode ser expresso por uma linguagem alvo denotado por $K_{locj} = E_j \parallel L_m(G_{locj})$. A partir disso, obtêm-se os supervisores modulares locais S_{locj} , $j = 1, \dots, n$, calculando as máximas linguagens controláveis locais $S_{up}C(K_{locj}, G_{locj})$, sendo n o número de supervisores locais, que neste exemplo, são três, os quais atuando em conjunto coordenam o comportamento do sistema.

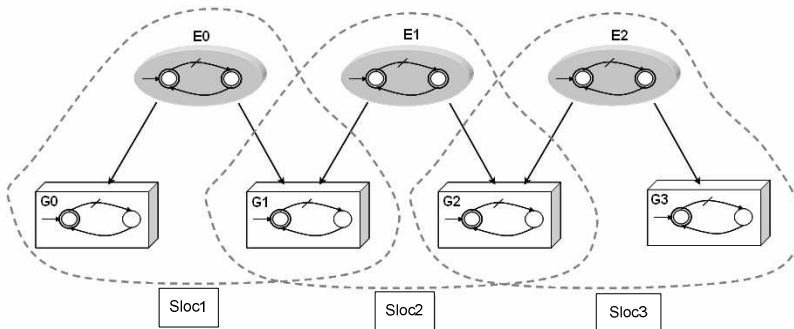


Figura 3.3 Exemplificação para obtenção de supervisores locais

Não obstante, deseja-se que o comportamento conjunto e concorrente dos supervisores modulares tenha a mesma ação de controle que a de um supervisor monolítico, ou seja, é necessário assegurar a modularidade local do conjunto de supervisores locais, garantindo que a ação

conjunta de todos os supervisores seja não bloqueante, conforme demonstrado por Queiroz, et. al, (2001). A verificação da modularidade local consiste em realizar a composição síncrona de todos os supervisores modulares locais, chamado de teste de modularidade. Caso o gerador resultante dessa composição não tenha estados bloqueantes, ou seja, é *trim*, pode-se afirmar que os supervisores locais são modulares entre si. Caso contrário, há conflito entre os supervisores e, como consequência, não é mais garantida a propriedade de não bloqueio dos supervisores modulares locais. Para resolver o conflito pode-se descartar a solução modular e calcular um supervisor monolítico; como também o uso de um supervisor monolítico em conjunto com os supervisores locais que atue como coordenador dos supervisores que estão gerando o conflito (QUEIROZ, 2004).

Como pode ser observado, a solução modular local é vantajosa em relação à abordagem monolítica, por aproveitar tanto a modularidade das especificações quanto da planta. Isso torna o projeto do sistema de controle mais flexível, já que permite a alteração de um supervisor local separadamente.

Entretanto, conforme destacado por Queiroz (2000), a complexidade do teste da modularidade local, por exigir a composição de todos os supervisores locais, acaba crescendo exponencialmente com o número de especificações e subsistemas envolvidos, visto que o problema de conflito é um problema global que ocorre pela interação de todos os supervisores com toda a planta. Além disso, a verificação da modularidade local será necessária sempre que uma modificação for feita no sistema de controle. Novos métodos mais eficientes têm sido desenvolvidos para que a modularidade possa ser verificada em sistemas complexos (PENA et al, 2009).

3.3 CÉLULA FLEXÍVEL DE MANUFATURA

Nesta seção é exemplificada a síntese de controle supervísório para um problema de controle de uma célula flexível de manufatura (CFM) conforme resolvido por Silva e Queiroz (2009) e Klinge (2007).

A CFM é uma bancada experimental que pertence ao Laboratório de Automação Industrial (LAI) do Departamento de Automação e Sistemas (DAS) da Universidade Federal de Santa Catarina, sendo uma plataforma para simulação de processos automatizados.

3.3.1 Descrição do sistema

A CFM, Figura 3.4, é composta por quatro estações operacionais que são conectadas através de uma mesa giratória, operando no sentido horário, e um manipulador robótico. Estas estações realizam a furação, a soldagem, o teste de qualidade e o retrabalho de peças. A CFM foi projetada para realizar a furação e soldagem de peças que chegam ao *buffer* de entrada. Porém, quando as peças não são fabricadas corretamente têm que ser reparadas. Assim, a estação de retrabalho funciona como entrada de peças manualmente restauradas.

A retirada de peças da mesa giratória pelo manipulador robótico se dá após o resultado do teste de qualidade. Três destinos são possíveis para o depósito de cada peça: armazém de aprovadas, de reprovadas e de peças danificadas, que necessitam de retrabalho.

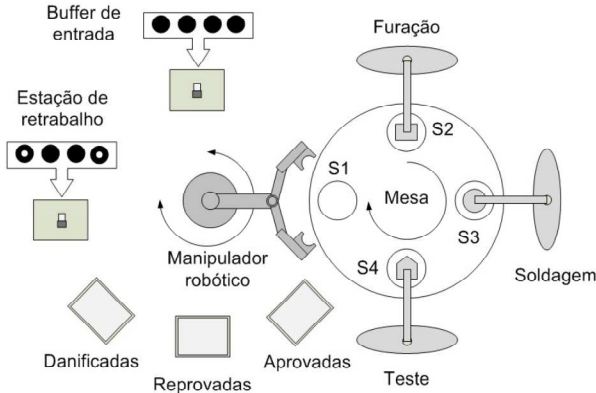


Figura 3.4 Célula flexível de manufatura (SILVA, 2010)

Conforme o comportamento proposto para a CFM, as peças que chegam ao *buffer* de entrada devem ser transportadas pelo manipulador robótico até a mesa giratória, na posição S1. Uma vez ali, ela é conduzida por meio da mesa giratória às estações de furação, soldagem e teste de qualidade onde as peças são testadas e retornadas à posição S1. Se o teste for positivo, as peças devem ser depositadas no armazém de peças aprovadas, caso contrário, no armazém de peças danificadas. Estas últimas serão restauradas na estação de retrabalho. Após serem retrabalhadas, as peças entram no sistema novamente por esta estação e o manipulador robótico as deposita na mesa giratória, na posição S1. Estas peças não devem ser furadas nem soldadas, só precisam ser testadas. Peças aprovadas devem ser depositadas no armazém de aprovadas. Se uma

peça é reprovada pela segunda vez, esta não será restaurada novamente e é depositada no armazém de peças reprovadas.

A CFM apresenta sensores para indicar a chegada de peças nas estações de furação, soldagem e teste. Porém, a modelagem do sistema é feita para que a presença de peças sobre a mesa seja deduzida a partir do histórico de eventos (SILVA, 2010).

Segundo Silva (2010), a CFM foi projetada para poder fabricar peças utilizando o limite máximo dos *slots* da mesa simultaneamente. Para atingir este objetivo, devem-se evitar alguns problemas que poderiam acontecer durante a fabricação de várias peças ao mesmo tempo, como:

- Girar a mesa nos casos seguintes: se não houver pelo menos uma peça sobre ela; ou enquanto qualquer das estações estiver realizando uma tarefa; ou caso existam peças esperando para serem furadas, soldadas, testadas ou removidas pelo manipulador robótico;
- Efetuar as operações de furação, soldagem, teste e manipulação robótica se não houver peça esperando pela realização das tarefas citadas no *slot* correspondente;
- Executar a mesma operação por duas vezes consecutivas em uma peça;
- Furar e soldar peças que chegam da estação de retrabalho;
- Sobrepor peças na posição S1;
- Acionar o manipulador robótico para pegar peças nas entradas do sistema sem haver peça nas mesmas.

3.3.2 Modelagem da planta

A modelagem da planta tem como objetivo a obtenção dos modelos dos subsistemas que representam a planta a ser controlada. Esta etapa começa com a identificação do conjunto de subsistemas envolvidos no problema, que no caso da CFM são: o manipulador robótico, *buffer* de entrada, estações de retrabalho, furação, soldagem e teste de qualidade. Em seguida, os subsistemas são modelados separadamente por geradores, cuja estrutura de controle é definida mediante a identificação de eventos controláveis e não controláveis. Na tabela 3.1, são apresentados os eventos utilizados para a modelagem dos geradores dos subsistemas e das especificações para a CFM. Observa-se que os eventos têm dois nomes, o primeiro em inglês e o segundo em português. Este último foi

utilizado na implementação do controle supervísório em CLP, a qual fará parte do conteúdo do capítulo 4. Os modelos dos geradores dos subsistemas da CFM são visualizados na Figura 3.5

Tabela 3.1 Eventos da Célula Flexível de Manufatura

G	Equipamento	Evento	Descrição
1	Sensor do <i>Buffer</i> de Entrada	sen_ib/asen	Sinaliza a chegada de peça no buffer de entrada.
2	Sensor da Estação de Retrabalho	sen_rw/asenret	Sinaliza a chegada de peça através da estação de retrabalho.
3	Manipulador Robótico	st_grabib/acolm	Pega peça do <i>buffer</i> de entrada e coloca na posição S1.
		fi_grabib/bcolm	Sinaliza o fim desta operação.
		st_grabr/acolmre	Pega peça da estação de retrabalho e coloca na posição S1.
		fi_grabr/bcolmre	Sinaliza o fim desta operação.
		st_putgood/aretok	Retira peça da posição S1 e coloca no compartimento de peças aprovadas.
		fi_putgood/bretok	Sinaliza o fim desta operação.
		st_putbad/aretnok	Retira peça da posição S1 e coloca no compartimento de peças reprovadas.
		fi_putbad/bretnok	Sinaliza o fim desta operação.
		st_putdmd/aretret	Retira peça da posição S1 e coloca no compartimento de peças danificadas.
fi_putdmdg/bretret	Sinaliza o fim desta operação.		
4	Mesa Giratória	st_turn/ames	Gira a mesa em 90 graus no sentido horário.
		fi_turn/bmes	Sinaliza o fim do giro da mesa.
5	Furadeira	st_drill/afur	Inicia a furação.
		fi_drill/bfur	Sinaliza o fim da furação.
		st_skdrill/apfur	Não realiza a furação.
		fi_skdrill/bpfur	Sinaliza a não realização da furação.
6	Soldadeira	st_weld/asol	Inicia a soldagem.
		fi_weld/bsol	Sinaliza o fim da soldagem.
		st_skweld/apsol	Não realiza a soldagem.
		fi_skweld/bpsol	Sinaliza a não realização da soldagem.
7	Teste	st_tes1/ates1	Inicia o teste das peças brutas.
		st_tes2/ates2	Inicia o teste das peças retrabalhadas.
		fi_ok/bok	Sinaliza a aprovação no teste.
		fi_nok1/bnok1	Sinaliza a primeira rejeição no teste.
		fi_nok2/bnok2	Sinaliza a segunda rejeição no teste.

Dado que os problemas de controle mencionados na seção 3.3.1 relatam situações que poderiam ocorrer durante a interação dos subsistemas, estes podem ser modelados em forma abstrata, envolvendo basicamente

o início e fim de operações, o que não interfere no desempenho do controlador (SILVA, 2010).

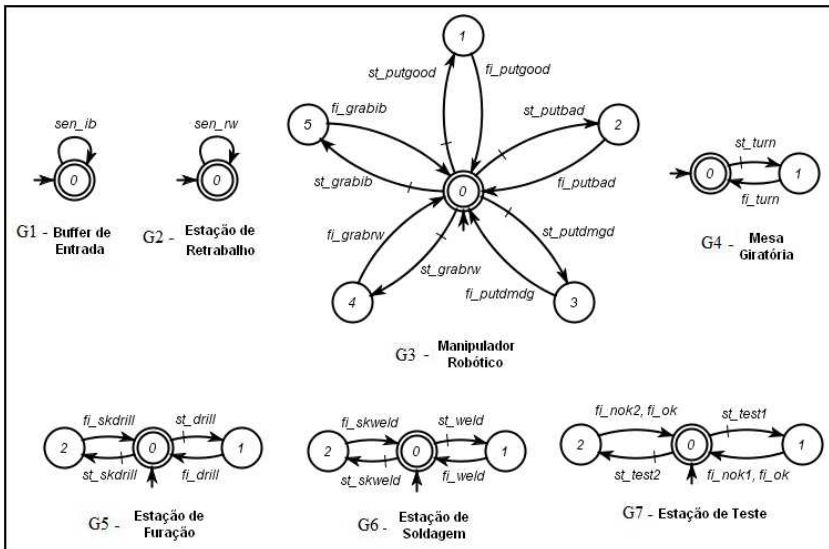


Figura 3.5 Geradores para os subsistemas da CFM (SILVA, 2010)

O *buffer* de entrada (G1) e a estação de retrabalho (G2) têm sensores que sinalizam a chegada de peças através dos eventos não controláveis *sen_ib* e *sen_rw* respectivamente, os quais podem ocorrer em qualquer momento. O manipulador robótico executa 5 diferentes ações. Ele pode pegar peças do *buffer* de entrada, *st_grabib* ou da estação de retrabalho, *st_grabrw* e depositar as peças nos três compartimentos de armazenamento: de peças aprovadas, *st_putgood*, reprovadas, *st_put_bad* e danificadas, *st_putdmgd*. A mesa giratória (G4) começa a girar pelo comando *st_turn* e quando termina a rotação, o sinal *fi_turn* é enviado. As estações de furação (G5) e soldagem (G6) funcionam de forma semelhante à mesa giratória, exceto para a possibilidade de não realizar as operações de furação e soldagem para peças vindo da estação de retrabalho pelos eventos *st_skdirill* e *st_skweld*, assim conforme foi mencionado na seção 3.3.1. Um modelo diferente foi proposto para o teste para avisar ao sistema quantas vezes uma peça foi rejeitada. Assim, a estação de teste (G7) executa, logicamente, dois testes diferentes. O teste 1, iniciado através do evento *st_test1*, é aplicado a peças provenientes do *buffer* de entrada, e tem como resposta uma aprovação, mediante o evento *fi_ok* ou a primeira rejeição com *fi_nok1*. O teste 2, iniciado

atrav3s do evento st_teste2 3 para aquelas pe3as que chegam da esta33o de retrabalho e o seu resultado indica uma aprova33o pelo evento fi_ok ou a segunda rejei33o pelo evento fi_nok2 .

3.3.3 Modelagem das especifica33es

As especifica33es modelam o comportamento desejado para a planta, mediante a constru33o de geradores para cada restri33o de coordena33o do sistema a ser controlado. Para a CFM, foram modeladas 11 especifica33es que podem ser visualizadas na Figura 3.6, as quais resolvem os problemas de intera33o entre os subsistemas, mencionados na se33o 3.3.1.

A especifica33o E1 preocupa-se em n3o permitir que a mesa gire sem ter pe3as em algum de seus *slots*. Isto 3 feito permitindo o comando st_turn , somente ap3s a conclus3o de a33es que garantem pelo menos a exist3ncia de uma pe3a sobre a mesa, ou seja, ap3s ocorrer pelo menos um dos eventos fi_grabrw , fi_grabib , fi_drill , $fi_skdrill$, fi_weld , fi_skweld , fi_ok , fi_nok1 ou fi_nok2 . Essa especifica33o cont3m um auto-la3o no estado 2. Isso significa a habilita33o dos eventos mencionados anteriormente mais de uma vez e assim a possibilidade de executar estas tarefas concorrentemente.

Para lidar com a exclus3o m3tua que deve existir entre a mesa e as esta33es, foram modelados quatro geradores, E_{2a} , E_{2b} , E_{2c} e E_{2d} . A partir delas pode-se afirmar que uma vez a mesa est3 girando, n3o deve haver a execu33o de tarefas pela respectiva esta33o, at3 que o evento fi_turn seja sinalizado e vice-versa.

As especifica33es E_{3a} , E_{3b} , E_{3c} e E_{3d} cuidam da sincroniza33o entre as esta33es. Cada gerador foca na resolu33o de coordena33o entre duas esta33es sequ3nciais. Os estados dos geradores s3o definidos por dois n3meros que indicam a presen3a de uma pe3a num *slot*, se diferente de 0, e os n3meros 1, 2 e 3 indicam uma qualidade para a pe3a. Assim, nas especifica33es E_{3a} , E_{3b} e E_{3c} , o n3mero 1 simboliza pe3as vindas do *buffer* de entrada e, o n3mero 2 pe3as vindas da esta33o de retrabalho. O teste pode resultar em tr3s respostas. Portanto, os n3meros 1, 2 e 3 dos estados da especifica33o E_{3d} representam uma aprova33o, uma primeira rejei33o e uma segunda rejei33o, respectivamente.

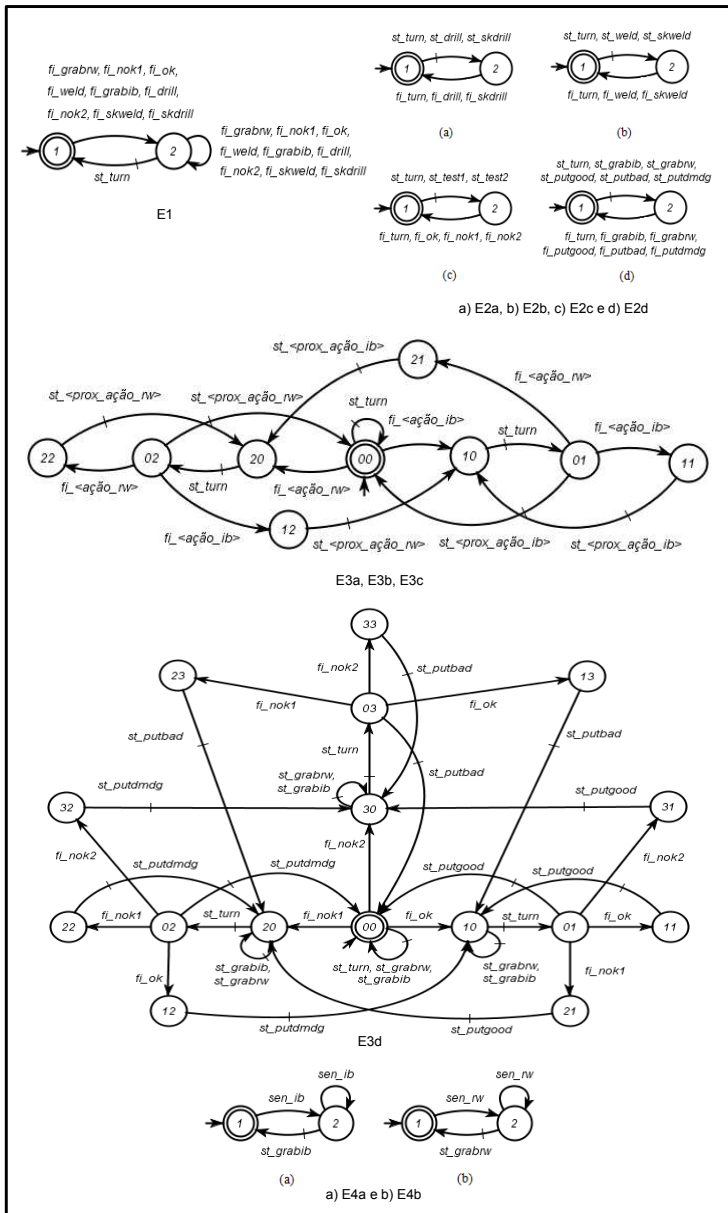


Figura 3.6 Especificações para a CFM (SILVA, 2010)

Os eventos $fi_<ação_ib>$ nas especificações E_{3a} , E_{3b} e E_{3c} correspondem aos sinais de fim de operação do subsistema anterior sobre uma peça vinda do *buffer* de entrada e, os eventos $st_<prox_ação>$ representam o início das operações sobre as peças vindas do *buffer* de entrada pelo subsistema seguinte ao que sinalizou o evento $fi_<ação_ib>$. O mesmo aplica-se aos eventos $fi_<ação_rw>$ e $st_<prox_ação_rw>$ em relação às peças provenientes da estação de retrabalho.

Por exemplo, em E_{3b} $fi_<ação_ib>$ é fi_drill , $prox_ação_ib$ é st_weld , $fi_<ação_rw>$ é $fi_skdrill$ e $prox_ação_ib$ é fi_skweld . A especificação E_{3a} responde pelo manipulador robótico, *slot* S1, e a estação de furação e E_{3c} pelas estações de soldagem e de teste. A especificação E_{3d} responde pela estação de teste e o manipulador robótico e segue o mesmo padrão de construção das especificações E_{3a} , E_{3b} e E_{3c} , mas difere em tamanho. Isso acontece porque a estação de teste oferece três possibilidades de resposta fi_ok , fi_nok1 e fi_nok2 . Por último, as especificações E_{4a} e E_{4b} estabelecem que o manipulador robótico deposita peça na mesa giratória após a ativação dos sensores sen_ib e sen_rw , respectivamente. O auto-laço no estado 2, permite a ativação dos sensores em qualquer momento. Para os leitores que desejam maiores detalhes da modelagem da CFM podem ser encontrados em Silva (2010).

3.3.4 Síntese de supervisores modulares locais da CFM

A síntese de supervisores para a CFM foi feita usando a abordagem modular local, apresentada na seção 3.2, onde cada um dos supervisores locais representa a máxima linguagem controlável para cada planta local. Dessa forma, cada supervisor local coordena uma parte do sistema global.

Conforme foi apresentado na seção 3.2, deve-se obter as plantas locais para cada especificação, compondo-se os subsistemas que possuem eventos em comum com a especificação em questão. Por exemplo, a construção da planta local G_{loc2a} , referente à especificação E_{2a} , com o alfabeto $\Sigma_{E_{2a}} = \{st_turn, st_drill, st_skdrill, fi_turn, fi_drill, fi_skdrill\}$, obtém-se mediante a composição síncrona dos geradores G_4 e G_5 , já que compartilham eventos com esta especificação, como pode ser obtido, a partir da tabela 3.1. As plantas locais restantes podem ser visualizadas na tabela 3.2. Na sequência, é obtida a linguagem alvo para cada planta local, através do produto síncrono da planta local com sua respectiva especificação, $K_{locx} = E_x \parallel (G_{locx})$, para $j \in \{1, 2a, 2b, 2c, 2d, 3a, 3b,$

3c, 3d, 4a, 4b}. Por último, é computada a máxima linguagem controlável contida em cada linguagem alvo, $S_{up}\mathcal{C}(K_{locx}, G_{locx})$. Para a CFM foram obtidos 11 supervisores locais, S_{locx} , $x \in \{1, 2a, 2b, 2c, 2d, 3a, 3b, 3c, 3d, 4a, 4b\}$. Em seguida, foi verificada a modularidade, calculando o gerador $S = S_{loc1} \parallel S_{loc2a} \parallel S_{loc2b} \parallel S_{loc2c} \parallel S_{loc2d} \parallel S_{loc3a} \parallel S_{loc3b} \parallel S_{loc3c} \parallel S_{loc3d} \parallel S_{loc4a} \parallel S_{loc4b}$ com 19180 estados. Dado que o gerador S é *trim*, diz-se que comportamento obtido pela supervisão das plantas locais é não bloqueante e equivalente ao comportamento obtido por supervisão monolítica. O número de estados de todos os geradores envolvidos na síntese de supervisores modulares locais é visualizado na tabela 3.3. Percebe-se que, apesar de ter sido projetado um controle modular, alguns supervisores possuem um número de estados bem elevado, como o S_{loc1} , que tem 648 estados. Isso torna sua implementação pouco prática e, portanto, faz sentido tentar reduzir os supervisores obtidos. Estes supervisores modulares reduzidos terão um número menor ou igual de estados, porém possuem a mesma ação de controle sobre o comportamento da planta livre local, que os correspondentes supervisores modulares locais. Um algoritmo formal para minimização de supervisores é apresentado em Su e Wonham (2004). A síntese dos supervisores locais e sua posterior redução foram realizadas na ferramenta computacional TCT (seção 3.5.2).

Tabela 3.2 Plantas locais para a CFM (SILVA, 2010)

G_{locx}	$\parallel G_x$
G_{loc1}	$G_4 \parallel G_5 \parallel G_6 \parallel G_7 \parallel G_3$
G_{loc2a}	$G_4 \parallel G_5$
G_{loc2b}	$G_4 \parallel G_6$
G_{loc2c}	$G_4 \parallel G_7$
G_{loc2d}	$G_4 \parallel G_3$
G_{loc3a}	$G_4 \parallel G_5 \parallel G_3$
G_{loc3b}	$G_4 \parallel G_5 \parallel G_6$
G_{loc3c}	$G_4 \parallel G_6 \parallel G_7$
G_{loc3d}	$G_4 \parallel G_7 \parallel G_3$
G_{loc4a}	$G_1 \parallel G_3$
G_{loc4b}	$G_2 \parallel G_3$

Tabela 3.3 Número de estados dos geradores da síntese de supervisores (SILVA, 2010)

X	E_x	G_{locx}	E_{locx}	S_{locx}	RS_{locx}
1	2	324	648	648	2
$2_a, 2_b, 2_c$	2	6	4	4	2
2_d	2	12	7	7	2
3_a	9	36	324	252	9
$3_b, 3_c$	9	18	162	90	9
3_d	16	36	576	288	16
$4_a, 4_b$	2	6	12	12	2

3.4 ARQUITETURA DE CONTROLE SUPERVISÓRIO

De acordo com Ramadge e Wonham (1989) uma arquitetura de controle constitui-se de duas estruturas: uma chamada planta, que gera os eventos de forma espontânea e assíncrona no tempo, e a outra constituída pelos supervisores, os quais exercem a ação de controle sobre o comportamento livre da planta. Porém, nos sistemas de manufatura, a maioria de eventos controláveis são executados através de comandos enviados pelo sistema de controle. Todos os eventos, controláveis e não controláveis, devem estar associados as suas próprias sequências operacionais, uma vez que não representam entradas diretas ou saídas do CLP.

Com o objetivo de implementar a abordagem modular local, numa estrutura organizada, com supervisores reduzidos, o sistema de controle é programado em três níveis hierárquicos (Queiroz e Cury, 2002), Figura 3.7, de modo que há uma interface entre os supervisores modulares e o sistema real: o sistema produto e as sequências operacionais. Esta arquitetura, cuja dinâmica é explicada a seguir, pode ser implementada em linguagem de Controladores Lógicos Programáveis (IEC 61131-3, 1998), em linguagens de PC (C++, Java, etc.) ou mesmo diretamente em hardware (circuito elétrico, pneumático ou hidráulico) (Queiroz e Cury, 2002).

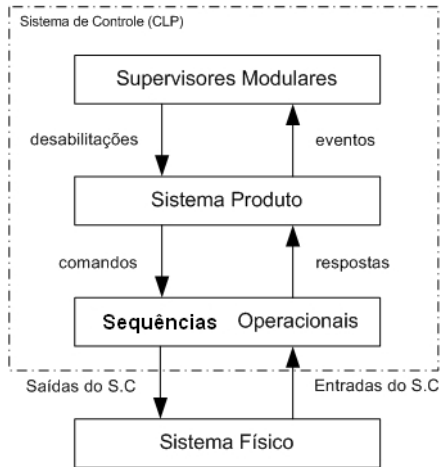


Figura 3.7 Arquitetura de Controle Supervísório (QUEIROZ e CURY, 2002)

A ação de controle no nível dos Supervisores Modulares (SM) se baseia nos estados ativos dos supervisores, que são atualizados de acordo com a ocorrência de eventos gerados na planta. Sendo assim, em cada estado ocorrerá uma ação de desabilitação de eventos controláveis que não serão permitidos de ocorrer na planta.

O nível do Sistema Produto (SP) tem como principal função executar os comandos que são permitidos e não são desabilitados pelo supervisor e, receber respostas enviadas pelas sequências operacionais, sinalizando, então, as mudanças de estado para os supervisores. Os supervisores são atualizados, toda vez que o SP envia comandos ou recebe respostas.

O nível Sequências Operacionais (SO), são procedimentos de baixo nível, que traduzem os comandos dos modelos abstratos do SP para gerar os sinais de saída do sistema de controle (entradas do sistema real) e por sua vez, lê os sinais de entrada (saídas do sistema real), gerando respostas lógicas ao SP, associadas a eventos não controláveis.

3.5 FERRAMENTAS PARA SIMULAÇÃO E IMPLEMENTAÇÃO

Durante o processo de síntese dos supervisores, a utilização de ferramentas computacionais como TCT (WONHAM, 1999), SUPRE-MICA (AKESSON, 2002), IDES (GRIGOROV, 2007), I-DES2ST(KLINGE, 2007), DESCO (FABIAN; HELLGREN, 2000),

GRAIL (RAYMOND e WOOD, 1994, CURY, 2001, REISER, 2005) UKDES (CHANDRA et al., 2002) e VER (BALEMI et al., 1993) torna esta metodologia de projeto diferenciada das tradicionais técnicas manuais e intuitivas, utilizadas como ferramentas nos trabalhos atuais de projeto lógico. Existem, naturalmente, mais programas ou ferramentas, que tratam os sistemas a eventos discretos. Dado que nesta dissertação foram usadas apenas IDES, TCT, Supremica e Ides2St, só serão introduzidas estas quatro ferramentas.

3.5.1 IDES

IDES é uma ferramenta fácil de usar, baseada em Java e desenvolvida pelo laboratório DES da Universidade de Queen. Ela permite uma aproximação visual permitindo criar e editar geradores por meio de um mouse. Também permite realizar operações sobre os geradores como composição síncrona, minimização, cálculo da máxima linguagem controlável, cálculo do *trim*, entre outras e, importar e exportar arquivos GRAIL e TCT (GRIGOROV, 2007).

3.5.2 TCT

TCT foi desenvolvida pelo Grupo de Controle de Sistemas da Universidade de Toronto no Canadá. É uma ferramenta para síntese de controle supervísório para sistemas a eventos discretos. Permite a criação de modelos para geradores e realizar com estes, operações como produto síncrono, calcular a máxima linguagem controlável e os eventos desabilitados de um supervisor, reduzir supervisores, entre outras. Para Windows e Linux, a ferramenta pode ser descarregada da página web do professor W. M. Wonham (FENG e WONHAM, 2006).

3.5.3 Supremica

Supremica é uma ferramenta para verificação, simulação e síntese de controladores para sistemas a eventos discretos. Supremica foi criada inicialmente como suporte ao trabalho de doutorado “Métodos e ferramentas em teoria de controle supervísório” do Knut Akesson da Universidade Técnica Chalmers na Suécia. Um projeto Supremica consiste de múltiplos geradores que juntos definem o problema principal. Esta ferramenta tem uma interface gráfica, a qual permite realizar facilmente a

síntese de supervisores e verificação, onde são testadas propriedades do sistema por meio de métodos formais. Além disso, possui uma interface para simulação do comportamento das plantas sob ação dos supervisores, o que permite avaliar a fase de modelagem e síntese de supervisores de um sistema de controle (AKESSON, 2002).

3.5.4 Ides2ST

Ides2ST é uma ferramenta desenvolvida por Klinge (2007), a qual gera automaticamente código em linguagem de texto estruturado (IEC-61131-3) da estrutura de controle hierarquizada apresentada na seção 3.4. A ferramenta precisa ter como entrada duas pastas. A primeira deve conter os modelos em IDES dos supervisores modulares reduzidos e, a segunda, os geradores do sistema produto também em IDES. Se o controle supervísório foi sintetizado a partir GRAIL ou TCT, os arquivos nessas ferramentas podem ser importados a IDES e depois convertidos em ST (KLINGE, 2007). Na Figura 3.8, é apresentado um exemplo da estrutura de código de implementação em Ides2ST.

Ides2St atribui uma variável de memória do CLP a cada gerador e a cada evento. Durante a inicialização do programa, todos os supervisores e subsistemas são definidos para seus estados iniciais, Figura 3.8 (a), se a variável auxiliar *ilc_inited* não está setada. Isto, para garantir que a lógica de inicialização seja executada só uma vez, dado que a varredura de um programa de CLP é sequencial. Por outro lado, Ides2St utiliza a variável auxiliar *evt_blk*, que quando ativa, significa uma atualização de estado no sistema produto e nos supervisores, para evitar a situação indesejável de computar duas transições subsequentes sem atualizar os supervisores. Isso não significa que os eventos subsequentes serão ignorados pelo programa, mas adiados. Suas variáveis são definidas em 1, mas elas são executadas em um estado conveniente, nos supervisores.

O código do programa relacionado ao nível de supervisores é dividido em duas partes. A primeira, Figura 3.8 (b), especifica os supervisores como geradores, os quais descrevem o estado atual e o evento de transição que leva para o próximo estado. Cada supervisor é composto por um grupo de instruções tipo IF que definem o número do estado atual e o número do estado alvo quando uma transição ocorre. A segunda parte, Figura 3.8 (c), refere-se à desabilitação de eventos que é feita através da definição de uma variável chamada *De_<evento>*. Esta parte é composta por instruções tipo CASE no qual a variável *De_<evento>* é

definida como 1 quando os estados das variáveis dos supervisores que desabilitam esse eventos estão ativos.

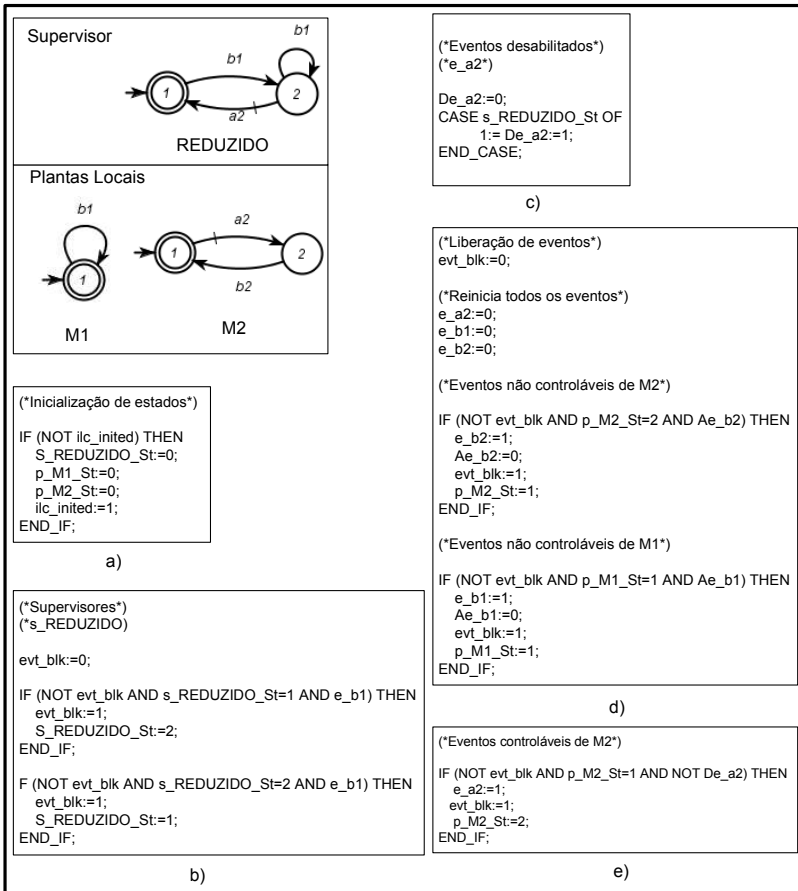


Figura 3.8 Exemplo da estrutura código de implementação em Ides2ST

O sistema produto também é implementado como geradores, semelhante aos supervisores, e seu código é dividido em duas partes. O código de eventos não controláveis, Figura. 3.8 (d), e eventos controláveis, Figura. 3.8 (e). A primeira parte deve preceder a última em consequência da sua imprevisibilidade. Assim, antes que qualquer evento controlável possa ser acionado, o código irá verificar se há um bit de um evento não controlável definido como 1. Se assim for, o subsistema é atualizado e a variável do evento de resposta $e_{<evento>}$ é definido como 1, para atualizar o supervisor. Se não fosse assim, e o subsistema

está em um estado no qual um evento controlável não está desabilitado, o bit da variável $e_{\langle evento \rangle}$ é definido como 1, ou seja, um comando é enviado para iniciar uma sub-rotina e para atualizar o supervisor, após este subsistema o faz. Quando um evento está definido em múltiplas transições do mesmo gerador, pode acontecer que o processamento da primeira transição desative o sinal de ocorrência desse evento, mesmo que a transição não esteja ativa. Para evitar este bloqueio nas transições do sistema produto, são zeradas todas as variáveis $e_{\langle evento \rangle}$ e a variável evt_blk , no início do código do sistema produto / final dos supervisores.

No nível de sequências operacionais, Ides2ST envia os comandos para iniciar a execução de procedimentos de baixo nível associados ao sistema de controle, assim como o encaminhamento de respostas para o sistema produto. Ides2ST deixa um espaço de memória em branco para ser associado com as sub-rotinas de início e fim implementadas pelo projetista. Quando uma sub-rotina termina, ele define como 1 o bit da variável $Ae_{\langle evento \rangle}$.

3.6 CONCLUSÃO DO CAPÍTULO

A Teoria de Controle Supervisório usada para Sistemas a Eventos Discretos permite a modelagem de subsistemas de manufatura e seus respectivos supervisores mediante a abordagem Modular Local que será objeto de utilização no desenvolvimento deste trabalho, a qual possibilita a utilização de elementos de controle reduzidos, uma vez que considera plantas locais, que envolvem um número menor de subsistemas. Ao invés de um único supervisor, obtêm-se vários supervisores menores, sendo um para cada especificação, mas que em conjunto possuem a mesma ação de controle, caso a sua modularidade seja verificada.

A implementação do sistema de controle, realiza-se em controladores lógicos programáveis, com base na arquitetura de controle supervisório, proposta por Queiroz e Cury (2002) constituída de três níveis hierárquicos: supervisores modulares, sistema produto e sequências operacionais. Por último, foram apresentadas as ferramentas computacionais que auxiliam na modelagem dos subsistemas da planta, simulação e validação da lógica de controle proposta para um sistema.

No capítulo 4, é introduzida formalmente a metodologia, objeto desde trabalho, terminando com a exemplificação da aplicação de funcionalidades do sistema SCADA na programação do controle supervisório.

rio da CFM estudada na seção 3.3. Esta arquitetura será utilizada para desenvolver o estudo de caso do capítulo 5.

4. METODOLOGIA PARA DESENVOLVIMENTO INTEGRADO DE SISTEMAS SCADA COM CONTROLE SUPERVISÓRIO

Este capítulo propõe uma metodologia para desenvolvimento de sistemas SCADA, aplicada a sistemas de manufatura, a qual é baseada na arquitetura de controle supervisão (QUEIROZ e CURY, 2002) explicada no capítulo 3 e implementada em dispositivos de controle CLP.

O capítulo está dividido em três seções: na seção 4.1, é descrita, passo a passo, uma metodologia de implementação de Sistemas de Supervisão e Aquisição de Dados integrada à programação do controle supervisão em CLP; na seção 4.2, é descrita a implementação de funcionalidades que oferece o sistema SCADA a partir do exemplo do capítulo 3 e finalmente na seção 4.3, são apresentadas as conclusões.

4.1 METODOLOGIA

Conforme uma metodologia adotada no trabalho de Silva e Queiroz (2009), a qual segue as seguintes etapas: modelagem da planta e especificações, síntese de supervisores, emulação de geradores e implementação segundo a ACS, é realizado um estudo de caso para explorar a integração de funcionalidades do sistema SCADA com o controle supervisão, dando como resultado uma implementação preliminar que dá origem à metodologia proposta neste capítulo.

A Figura 4.1 apresenta essa metodologia, que é desenvolvida em oito fases: projeto informacional; síntese de controle supervisão; emulação; implementação de controle supervisão em CLP; implementação de funcionalidades básicas do SCADA; avaliação de funcionamento do sistema real; implementação de funcionalidades gerais do sistema SCADA; e validação. Cada uma dessas fases é explicada a seguir.

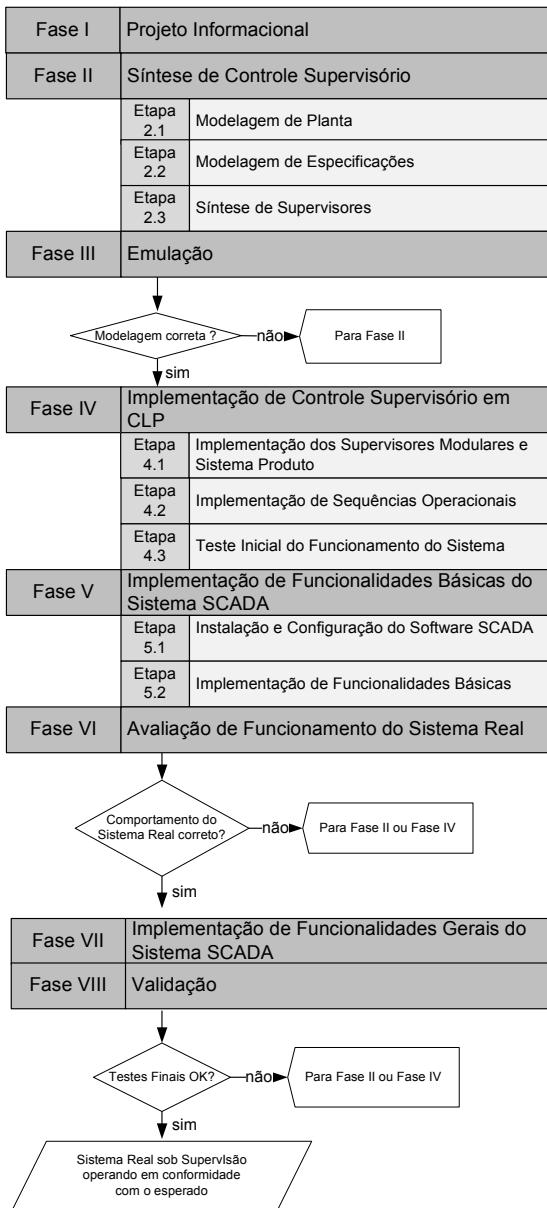


Figura 4.1 Metodologia

4.1.1 Fase I – Projeto informacional

O projeto informacional 3 a primeira fase da metodologia proposta. Aqui s3o levantadas as especifica33es t3cnicas do projeto de integra333o, mediante a obten333o das informa3333es relevantes do processo a ser controlado e supervisionado. Essas informa3333es facilitam an3lise e compreens3o do sistema e auxiliam ao projetista na fase de modelagem.

Nesta fase descreve-se o comportamento fisicamente poss3vel do sistema, identificando os subsistemas ou equipamentos que o comp33em, bem como o problema de controle em s3, definindo as restri3333es do comportamento e coordena333o desejados para o sistema. Tamb3m, definem-se os dispositivos de campo: sensores e atuadores, que devem fazer parte do sistema sob controle, mediante a implementa333o das sequ3ncias operacionais, as quais s3o programadas nos equipamentos de controle.

Outro fator importante 3 a defini333o das tecnologias a serem utilizadas no desenvolvimento do projeto. Esta etapa envolve os equipamentos ou dispositivos que executam o controle autom3tico das atividades da planta, como por exemplo: CLP, RTU, microcontrolador, etc. Envolve ainda, a rede de comunica333o e protocolo a serem usados, para estabelecer a comunica333o entre o equipamento de controle e a esta333o de supervis3o. Apesar de serem v3rios os tipos de dispositivos de controle presentes nos sistemas de manufatura, esta metodologia est3 mais focada na implementa333o de CLP cuja programa333o 3 realizada em linguagens como *ladder* e texto estruturado (IEC-61131-3).

Por 3ltimo, esta fase define as ferramentas que permitem a s3ntese do controle supervisi3rio segundo a TCS; o software de configura333o e programa333o do CLP e o software que permite o desenvolvimento da aplica333o SCADA. Este 3ltimo, pode ser propriet3rio ou de c3digo aberto, desde que possua os m3dulos que implementem as funcionalidades apresentadas na se333o 2.5.1 e forne3am uma interface de comunica333o compat3vel com os equipamentos de controle e dispositivos de campo definidos para o projeto. O software SCADA deve possuir *drivers* de comunica333o compat3veis com os protocolos oferecidos pelo CLP.

4.1.2 Fase II – S3ntese de controle supervisi3rio

Esta fase compreende a modelagem da planta a ser controlada, modelagem das especifica3333es de controle, s3ntese de supervisores 3timos, teste de modularidade e redu333o de supervisores. As etapas de

modelagem devem atender os requisitos de projeto novo ou de re-projeto (melhoramento). O re-projeto se dá quando são necessárias modificações às leis de controle existentes ou quando são inseridos ou retirados equipamentos do sistema.

Etapa 2.1 – Modelagem da planta

Nesta etapa, são construídos os geradores que representam o comportamento independente de cada subsistema. É importante considerar na modelagem a maior abstração possível das atividades referentes aos modelos, considerando apenas os eventos relevantes que sejam necessários às tarefas de coordenação entre os subsistemas, para evitar a explosão do espaço de estados e facilitar o processo de síntese dos supervisores. Este procedimento resulta em modelos menores e, por consequência, em menor custo computacional.

Etapa 2.2 - Modelagem das especificações

Nesta etapa, realiza-se construção dos geradores de cada especificação isoladamente. As especificações determinam como a operação coordenada dos subsistemas deve ocorrer de acordo com o comportamento desejado pelo projetista. Possíveis alterações neste comportamento, decorrentes da inserção de novos produtos, de modificações em processos e produtos existentes, de novas necessidades de demanda, da modernização tecnológica, dentre outros, podem estar restritas à apenas uma especificação.

Etapa 2.3 - Síntese de supervisores

A etapa de síntese segue a abordagem modular local (QUEIROZ e CURY, 2000), apresentada no capítulo anterior. Primeiro, obtêm-se a planta local para cada especificação compondo-se os subsistemas que possuem eventos em comum com a especificação em questão; depois, calcula-se a especificação para cada planta local, através do produto síncrono da planta local com sua respectiva especificação; em seguida, calcula-se a máxima linguagem controlável contida em cada linguagem alvo; e por último, verifica-se a modularidade local (não conflito) dos supervisores resultantes; se forem modulares, implementar-se um supervisor local reduzido para cada linguagem controlável; se não forem modulares, resolver-se o conflito.

4.1.3 Fase III - Emulação

Na fase de emulação, são realizados testes da lógica de controle proposta, por meio da ferramenta definida na Fase I. A emulação permite acompanhar a atuação dos supervisores na planta sequencialmente, possibilitando identificar erros ocorridos na modelagem. Caso o projetista encontre erros de modelagem, permite-se realizar previamente mudanças no projeto, voltando à fase dois da metodologia proposta, caso contrário, os supervisores e o sistema produto podem ser implementados.

4.1.4 Fase IV - Implementação do controle supervisório em CLP

Esta fase é constituída de duas etapas, cujo objetivo é a implementação do sistema de controle conforme a hierarquia de três níveis: supervisores modulares, sistema produto e sequências operacionais, (QUEIROZ, CURY, 2002), a qual foi discutida na seção 3.4. Para atingir este propósito, cada nível é implementado através de um código no CLP.

Etapa 4.1 - Implementação dos supervisores modulares e sistema produto

Nesta etapa, são implementados na linguagem de texto estruturado os supervisores modulares, em suas versões reduzidas conforme apresentado na seção 3.3.4, e o sistema produto mediante a ferramenta de geração automática de código IDES2St (KLINGE, 2007), apresentada na seção 3.5. Esta ferramenta gera um código associado a cada nível, que é implementado através de módulos de procedimento no CLP.

Etapa 4.2 - Implementação de sequências operacionais

Nesta etapa, são implementadas as rotinas operacionais em qualquer linguagem de programação, mas suportada pelo CLP de acordo com IEC 61131 (*ladder*, texto estruturado, blocos funcionais, SFC, etc.). É recomendável, se possível, programar cada sequência operacional como uma máquina de estados, para tornar mais fácil a manutenção e entendimento do código implementado por parte dos projetistas e mantenedores dos sistemas.

Etapa 4.3 - Teste inicial do funcionamento do sistema

Nesta etapa é feita uma verificação inicial para encontrar possíveis inconsistências na lógica programada. Por conveniência no momento do teste, o projetista pode associar a cada sequência operacional rotinas de temporização que simulem o tempo de execução de operação de um determinado subsistema, pois nesta etapa ainda não foram inseridos os dispositivos de campo que são aqueles que implementam as sequências operacionais. Esta verificação pode não ser exaustiva, deixando para as fases seguintes a validação final da solução proposta.

4.1.5 Fase V - Implementação de funcionalidades básicas do sistema SCADA

Esta fase destina-se à implementação de quatro funcionalidades básicas fornecidas por um sistema SCADA. As funcionalidades chamadas de básicas são aquelas que fornecem informação gráfica e textual da planta que está sendo controlada a partir da programação de controle supervísório obtida na fase IV, permitindo interagir com o processo. Inclusive, tornam possível a visualização e análise de erros na modelagem e na programação das sequências operacionais que não puderam ser verificados nas fases II e IV respectivamente. Essas funcionalidades são: o sinótico, o envio de comandos, o histórico de eventos e a geração de alarmes críticos. Esta fase compreende duas etapas:

Etapa 5.1 – Instalação e configuração do software SCADA

Inicia-se com a configuração do servidor SCADA, responsável por armazenar e trocar informações com os clientes e onde está contido o BD do processo. Isto é realizado a través de uma chave de hardware ou *hardkey* em softwares proprietários, a qual é conectada ao computador servidor; ou mediante a instalação de um aplicativo como acontece, por exemplo, com o servidor de aplicações Tomcat integrado ao software ScadaBR. Feito isso, o software SCADA é instalado no servidor. Em seguida, é configurado o protocolo de comunicação tanto no CLP, quanto na aplicação SCADA, definindo assim, as áreas do CLP que poderão ser manipuladas através da rede de comunicação. Finalmente, é verificado o envio de dados entre eles até que a comunicação seja validada. A instalação do software não é necessária a cada novo projeto, sempre que

seja utilizado o mesmo computador servidor no qual foi implementada uma aplicação anterior.

Etapa 5.2 – Implementação de funcionalidades básicas

Nesta etapa, são implementadas funcionalidades como: o sinótico, envio de comandos, histórico de eventos e alarmes críticos no software SCADA, a partir de código gerado na programação do controle supervisório em CLP.

4.1.6 Fase VI - Avaliação de funcionamento do sistema real

Esta fase tem o propósito de avaliar se o comportamento do sistema real está de acordo com o inicialmente planejado. Aqui são realizados novos testes, desta vez, integrando os dispositivos de campo ao CLP. Se houver problemas de funcionamento e coordenação do sistema real, associados à modelagem incorreta e/ou programação de sequências operacionais inapropriadas, serão visualizados através das funcionalidades implementadas na etapa 5.2. Portanto, se o resultado dos testes é negativo, deve-se iniciar uma nova iteração da metodologia a partir da fase dois ou três, se for por modelagem ou programação de sequências operacionais, respectivamente. Não obstante, se o resultado dos testes é positivo, deve-se continuar com a execução da fase seguinte.

4.1.7 Fase VII - Implementação de funcionalidades gerais do sistema SCADA

Esta fase destina-se à implementação de funcionalidades gerais no software SCADA. As funcionalidades chamadas de gerais são aquelas que ampliam e complementam a informação gerada pelos processos executados no chão de fábrica, alimentando todas as áreas gestoras provendo base para conhecimento a áreas estratégicas como marketing, vendas e a financeira. Por exemplo, a informação gerada nesta fase pode auxiliar na tomada de decisões quando há falhas em algum equipamento, mudanças nas ordens de produção, inserção ou remoção de equipamentos, etc. As funcionalidades implementadas nesta fase são: geração de alarmes gerais, gráficos de tendências, receitas, relatórios e geração de informação para níveis gerenciais.

4.1.8 Fase VIII - Validação

Esta fase, a última da metodologia proposta, tem a finalidade de realizar os testes finais que permitam validar e certificar o funcionamento ótimo do sistema real e da aplicação SCADA para o sistema de manufatura. Caso o resultado seja negativo, será identificada a causa e da mesma forma como na fase VI, o projetista deverá iniciar uma nova iteração de metodologia a partir da fase II ou IV. No entanto, se o teste for positivo, a validação do sistema está completa e, portanto o sistema SCADA está implementado. Esta fase é realizada tantas vezes quantas forem necessárias, até conseguir que o sistema se comporte conforme o desejado.

4.2 IMPLEMENTAÇÃO DE FUNCIONALIDADES DO SISTEMA SCADA

Nas próximas seções, é descrita a implementação das fases cinco e sete da metodologia proposta utilizando a programação do controle supervísório da CFM apresentada no capítulo 3 e o software ScadaBR para o desenvolvimento da aplicação SCADA.

4.2.1 Sinótico

Um sinótico ou representação gráfica cumpre sua função quando o operador consegue acompanhar o comportamento e atuar de forma certa sobre o processo. Para alcançar esse propósito, é necessário um método para realizar uma interação humano-máquina de forma concisa e confiável.

Este método projeta o sinótico da planta a partir das variáveis definidas no código do CLP (Etapa 4.1) que representam os estados dos geradores dos supervisores e do sistema produto. Entretanto, é importante definir quais desses estados fornecem informação relevante de ser exibida: muita informação pode sobrecarregar o operador, e muito pouca, pode confundi-lo e induzi-lo a erros.

Para isso, realiza-se uma análise da semântica dos estados do sistema produto e dos supervisores, estabelecendo assim, quantos e quais estados são necessários e suficientes para visualizar corretamente o funcionamento da planta. Em seguida, estas variáveis são configuradas como *tags* no software SCADA, para ser inseridas na tela do sinótico

por meio de imagens ou objetos gr3ficos. Quando 3 necess3rio implementar combina33es ou condi33es entre vari3veis associadas 3 ocorr3ncia de eventos ou de estados, s3o configurados *tags* internos que s3o executados mediante *scripts* de programa33o. O sin3tico obtido por este m3todo permite dar clareza e confiabilidade ao operador sobre o comportamento e funcionamento do processo, conhecendo o estado atual do sistema.

A tela do sin3tico pode ser representada atrav3s de uma fotografia quando os processos s3o pequenos. Por3m, quando grandes 3 mais apropriado utilizar desenhos esquem3ticos que simbolizam de forma padronizada diversos processos industriais (motor, queimador, v3lvulas, bombas, etc). Os desenhos gr3ficos colocados sobre a tela do sin3tico bem como as cores utilizadas para ilustrar estados operacionais do processo, s3o de livre escolha pelo projetista/analista do sistema e tamb3m depende da pasta de gr3ficos disponibilizados pelo software SCADA. Por exemplo, uma forma gen3rica quanto ao uso de cores sobre a tela seria: verde piscando representando o sistema em opera33o, vermelho cont3nuo representando o sistema em repouso, azul piscando identificando a exist3ncia de pe3a em determinada posi33o e transparente identificando inexist3ncia de pe3a na referida posi33o.

- **Exemplo CFM**

A implementa33o do sin3tico da CFM (Figura 4.2) inicia-se com a an3lise do significado dos estados dos sete subsistemas que comp3em a planta. Por exemplo, o gerador G_7 da esta33o de teste, Figura 3.5, tem tr3s estados que podem ser assim descritos: o estado 0, significa em repouso; o estado 1, testando pe3a vinda do *buffer* de entrada; e o por 3ltimo, o estado 2, testando pe3a vinda da esta33o de retrabalho. Da mesma forma, a descri33o 3 feita para os demais geradores.

Ap3s a an3lise, conclui-se, que os estados dos geradores do sistema produto s3o necess3rios, mas n3o suficientes para representar graficamente, na tela do sin3tico, o comportamento geral da planta. Isso porque, a partir dessa informa33o n3o se consegue visualizar o percurso de uma pe3a de uma esta33o de trabalho a outra. Ent3o, s3o analisados os estados dos 11 supervisores, cujos significados revelam que os estados 3 e 4 dos supervisores 3A, 3B e 3C e os estados 4, 5 e 6 do supervisor 3D, tabela 4.1, fornecem essa informa33o. Observa-se que os supervisores reduzidos s3o muito semelhantes 3s especifica33es na Figura 3.6, o que facilita a compreens3o do significado de seus estados.

Tabela 4.1. Descrição dos estados relevantes dos supervisores para o sinótico da CFM

Supervisor	Estado	Significado
3A	3	Peça oriunda do <i>buffer</i> de entrada indo da posição S1 para S2.
3A	4	Peça oriunda da estação de retrabalho indo da posição S1 para S2.
3B	3	Peça oriunda da estação de retrabalho indo da posição S2 para S3.
3B	4	Peça oriunda do <i>buffer</i> de entrada indo da posição S2 para S3.
3C	3	Peça oriunda da estação de retrabalho indo da posição S3 para S4.
3C	4	Peça que chega do <i>buffer</i> de entrada indo da posição S3 para S4.
3D	4/5/6	Peça para retrabalho/aprovada/reprovada indo da posição S4 para S1.

Observa-se na Figura 4.2, o sinótico para a CFM que ilustra através de quadros brancos pontilhados onde é distribuída a informação coletada anteriormente. Os nomes G1, G2, G3, G4, G5, G6 e G7 identificam os geradores que representam os subsistemas da planta. Já os nomes S3A, S3B, S3C e S3D identificam os supervisores, apresentados na tabela 4.1.

Os círculos com letras *BE* e *ER* associados aos quadros G1e G2 ilustram a presença de uma peça oriunda do *buffer* de entrada ou da estação de retrabalho respectivamente. Estes círculos ou objetos gráficos aparecem quando o sistema está em funcionamento no espaço associado aos quadros S3A, G5, S3B, G6, S3C e G7. Para exemplificar esta situação, suponha que o sistema está em funcionamento e uma peça vinda do *buffer* de entrada chega à estação de furação (gerador G_5). Então, um círculo com letras *BE* aparece nesse lugar e o *led* que está em cima da furadeira pisca em cor verde quando está realizando a furação, associado ao estado 1 do gerador G_5 . Caso contrário, fica estático em cor vermelha. Uma vez terminada a furação, esta peça é conduzida à estação de soldagem (gerador G_6).

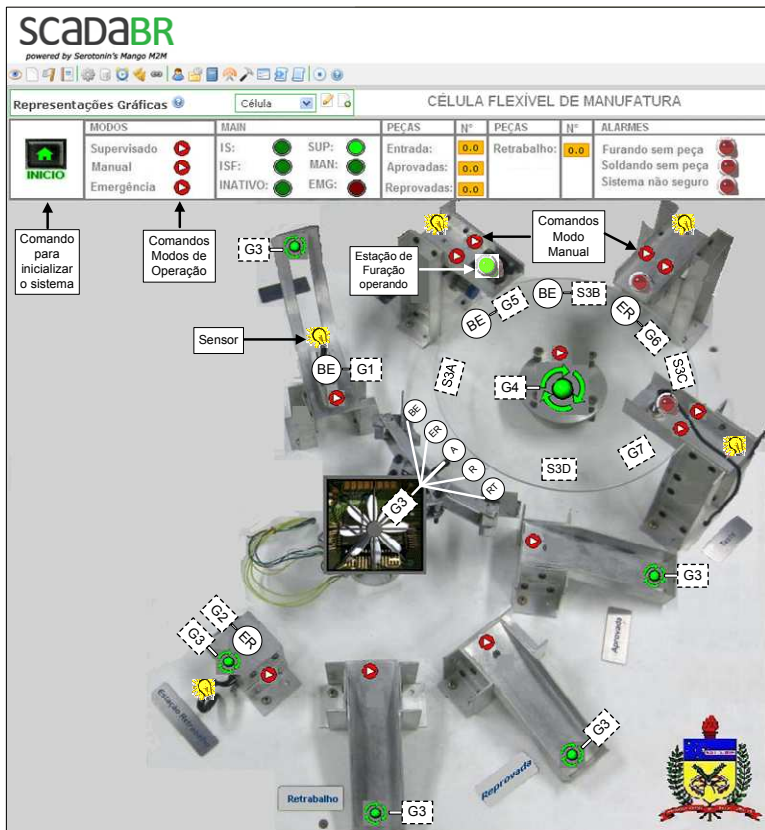


Figura4.2 Sinótico para a CFM

O percurso da peça entre as duas estações está representado pelo mesmo círculo associado ao estado 3 do supervisor S3B. A representação gráfica é a mesma para uma peça vinda da estação de retrabalho, só que com letras *ER*. Por outro lado, as ações do manipulador robótico (gerador G_3) estão representadas mais uma vez por círculos com letras *BE*, *ER*, *A*, *R*, e *RT*, ilustrando transporte de peças do *buffer* de entrada e da estação de retrabalho para o *slot* S1, e deste último para o depósito de aprovadas, reprovadas ou de retrabalho respectivamente. Além disso, os círculos verdes com flechas ficam girando quando o manipulador robótico está realizando suas operações. O círculo verde maior, associado ao estado 1 do gerador G_4 , também gira quando a mesa está funcionando. O

sinótico mostra ainda os sinais dos sensores e também exibe informação adequada do estado de operação do sistema de controle, os comandos que os ativam e os alarmes associados ao funcionamento geral do sistema por meio de objetos gráficos, os quais estão associados ao código em CLP que será discutido nas próximas seções.

4.2.2 Envio de Comandos

A Figura 4.3 caracteriza a estrutura do programa principal (SFC Main) proposta por Vieira *et al.*(2007), implementada em SFC (Sequential Function Chart) e a qual estabelece o modo de operação do sistema de controle. Este programa tem seis modos distintos de operação:

- *Software Initialization* (SI);
- *Physical System Initialization* (PSI);
- *Manual* (Man);
- *Supervised* (Sup);
- *Emergency* (Emg);
- *Idle* (Init e PSIted).

Considera-se que o passo *init* é ativado na primeira execução do programa. Quando o usuário ou operador ativa *Sinit*, o passo *SI* é ativado, significando que o programa está no modo de inicialização do sistema de controle. *PSinit* é uma entrada física que permite a inicialização do sistema, a ser controlado através do passo *PSI*. Quando a inicialização está pronta, a variável *PSReady* fica ativa, fazendo que o SFC evolua para o passo *PSIted*, conduzindo o sistema de controle ao modo *Idle*. Enquanto o sistema de controle estiver no modo de operação *Idle*, existem duas formas possíveis de se controlar o sistema físico: Uma é através da ativação do modo Manual (Man) e a outra forma é pelo modo Supervisionado (Sup).

No modo Manual, o operador conduz a realização das atividades a serem executadas pelo sistema físico, através de uma determinada sequência de eventos. Quando esta sequência está dentro do comportamento desejado, o sistema pode passar para o modo *Sup*. Caso contrário, o sistema deve passar logo para o modo *SI*. Já no modo Supervisionado (passo Sup ativado), a coordenação das ações a serem desenvolvidas pelo sistema físico é de acordo com o estado ativo do conjunto de supervisores. Isto é realizado através da ativação do sinal de desabilitação associado aos eventos controláveis. Por fim, a ativação do modo Emer-

gência (passo Emg) conduz o sistema físico a um estado apropriado de acordo com as características de cada processo e a conseqüente suspensão das atividades que até então estavam sendo realizadas.

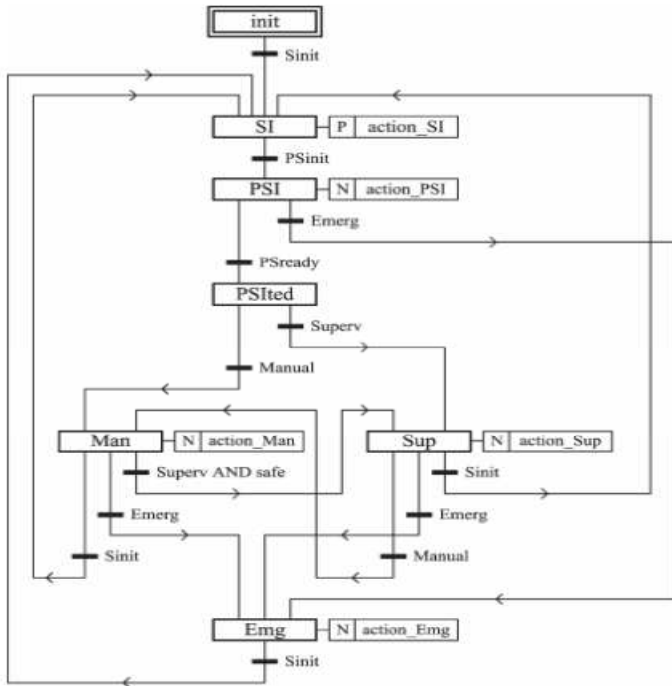


Figura 4.3 SFC Main (VIEIRA et al., 2007)

Esta estrutura é utilizada para exemplificar a funcionalidade envio de comandos. Ela é traduzida para a linguagem de texto estruturado, apenas no tocante à ativação dos modos a partir de botões no sinótico e especificamente no modo Manual, para forçar eventos controláveis.

Cabe ressaltar, que a estruturação do programa de controle, incluindo o programa Main de Vieira *et al.*(2007), tem por objetivo muito mais do que somente realizar o envio de comandos, ela visa implementar a arquitetura de controle proposta por Queiroz e Cury (2002), estabelecendo modos de operação ao sistema como um todo.

- **Exemplo CFM**

Para implementar o envio de comandos na CFM, é gerado um código em linguagem de texto estruturado, com instruções tipo IF, para ativar os modos anteriormente citados, conforme mostrado na tabela 4.2.

As variáveis booleanas *Sinit*, *PSInit*, *PSready*, *Superv*, *Manual* e *Emerg* são condições de transição (comandos) que ativam os diversos modos. As variáveis booleanas, *pausa* e *unsafe*, não são comandos e foram implementadas no código do sistema produto, para ser ativadas dependendo de certas condições. A ativação da variável *pausa* é usada no modo Manual, para não permitir a execução de eventos controláveis de forma espontânea, em razão de esta atividade ser realizada pelo operador. Já, a variável *unsafe* é ativada sempre para significar a violação de uma especificação de controle por parte do operador.

Além disso, no modo Manual foram criadas variáveis booleanas no código do sistema produto chamadas de *forca Equipamento*, sendo que a palavra *equipamento* está associada aos subsistemas que compõem a planta, para depois serem representadas graficamente em forma de botões no sinótico. Essas variáveis atuam como comandos, ativados pelo operador, para forçar a ocorrência de eventos controláveis no modo Manual. Por exemplo, *forca_es_sol* aciona a *soldadeira*, *forca_es_fur* aciona a furadeira e assim por diante para cada evento controlável associado a um equipamento da CFM. A explicação das variáveis nessa seção é apresentada na seção 4.2.4.

Os estados das variáveis *Sinit*, *Superv*, *Manual* e *Emerg*, também são determinados pelo operador no sinótico através de comandos como pode ser visto na Figura 4.2. Ao ser reconhecido estado de emergência pelo operador, o sistema deve ser reinicializado através do evento *Sinit*.

Tabela 4.2 Código em ST para implementar comandos

SI	PSI	PSIted
IF Sinit THEN SI:=1; São iniciadas com valor zero, todas as variáveis utilizadas no código do CLP, incluindo as variáveis de entrada e saída do CLP	IF PSinit THEN SI:=0; PSI:=1; Aqui, é executado o programa de início do sistema físico da CFM. Após execução, é ativada a variável PSready.	IF PSready THEN PSI:=0; PSIted:=1; PSready:=0; END_IF;

PSinit:=1; Sinit:=0; END_IF; IF (Sup OR Man OR Emg) AND Sinit THEN SI:=1; END_IF;	PSready:=1; PSInit:=0; END_IF;	
Man	Sup	Emg
IF PSIted AND Manual THEN PSready:=0; Man:=1; PSIted:=0; Manual:=0; pausa:=1; END_IF; IF Man AND NOT unsafe AND Superv THEN Sup:=1; Man:=0; Superv:=0; pausa:=0; END_IF; IF Man AND un- safe AND Superv THEN Man:=1; Superv:=0; END_IF;	IF PSIted AND Superv THEN PSready:=0; Sup:=1; PSIted:=0; Superv:=0; END_IF; IF Sup AND Manual THEN Man:=1; pausa:=1; Sup:=0; Manual:=0; ELSE IF Sup AND Emg THEN Emg:=1; Sup:=0; Emrg:=0; END_IF; END_IF;	IF Sup AND Emrg THEN Emg:=1; Sup:=0; Emrg:=0; END_IF; IF Man AND un- safe AND Emrg THEN Emg:=1; Man:=0; Emrg:=0; unsafe:=0; END_IF; Neste estado po- dem ser implemen- tadas, dependendo do processo, rotinas que levem a um estado seguro dos equipamentos.

4.2.3 Histórico de Eventos

Como foi mencionado no capítulo 2, um software SCADA permite de uma forma automática e a partir de uma base de dados, armazenar e registrar informação operacional, alarmes, gráficos de tendências, entre outros, através de históricos. Porém, nesta seção, é implementado outro histórico, de eventos controláveis e não controláveis que aconteceram num período de tempo específico, tornando possível conhecer a sequência de eventos ocorridos até o acontecimento de uma falha no sistema, e aqueles que continuavam na sequência, num processo normal

de operação. Esta lista de eventos complementa a informação fornecida pelos registros históricos anteriormente citados.

- **Exemplo CFM**

O histórico de eventos é implementado em cinco passos (Figura 4.4), tanto no CLP (passos 1 e 3) quanto no software SCADA (passos 2, 4 e 5), os quais são descritos a seguir.

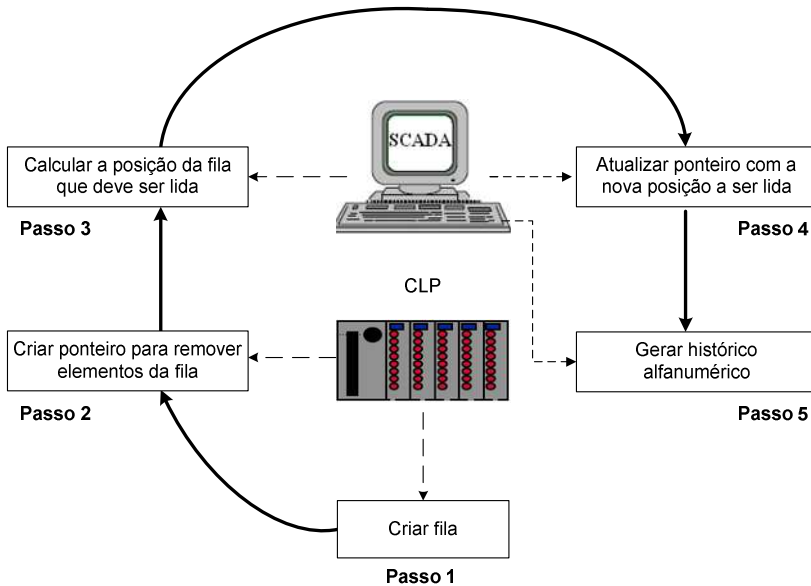


Figura 4.4 Implementação de histórico de eventos

Observa-se no passo 1, Figura 4.5, uma variável denominada *eventos*. Essa variável é inserida inicialmente no código que implementa o sistema produto. Ela é usada para armazenar o último evento ocorrido seja este controlável ou não controlável, identificado através de um número. Como exemplo, observa-se na Figura 4.6 que ao ocorrer o fim do giro da mesa, o evento *bmes* é ativado e o número 1 é escrito na variável *eventos*. Da mesma forma acontece quando ocorre o início do giro da mesa através do evento *ames*, só que agora o valor desta variável muda para 2.

```
(* Início código Supervisores*)
IF (evt_blk) THEN
  evt_blk := 0;
  num_eventos := num_eventos + 1;
  i:=num_eventos MOD 10;
  CASE i OF
    0 : fila_1 := eventos;
    1 : fila_2 := eventos;
    2 : fila_3 := eventos;
    3 : fila_4 := eventos;
    4 : fila_5 := eventos;
    5 : fila_6 := eventos;
    6 : fila_7 := eventos;
    7 : fila_8 := eventos;
    8 : fila_9 := eventos;
    9 : fila_10 := eventos;
  END_CASE;
END_IF;
....
```

Figura 4.5 Passo 1: Criação da fila

```
(*Sistema Produto*)
(*eventos não controláveis de p_G4:*)
IF ((NOT evt_blk) AND (p_G3_St=2) AND Ae_bmes) THEN
  e_bmes := 1;
  Ae_bmes := 0;
  evt_blk := 1;
  p_G3_St:=1;
  eventos := 1;
END_IF;
....
(*eventos controláveis de p_G4:*)
IF ((NOT evt_blk) AND (p_G3_St=1) AND NOT De_ames) THEN
  e_ames := 1;
  evt_blk := 1;
  p_G3_St:=2;
  eventos := 2;
END_IF;
....
```

Figura 4.6 Trecho de código do sistema produto

Posteriormente, a variável *eventos* é inserida no código que implementa os supervisores. Neste código, após a declaração de variáveis e a ativação da variável *evt_blk* (ocorrência de um evento no sistema produto), realiza-se a contagem de eventos através da variável *num_eventos*. Em seguida é implementada uma fila FIFO (*First-In-*

First-Out) onde os elementos (eventos) são inseridos e removidos de acordo com o princípio “o primeiro que entra é o primeiro que sai”. Os elementos da fila sempre são inseridos no final e removidos do começo.

Observa-se, na Figura 4.5, a existência da variável *i*. Essa variável é um índice que indica a próxima posição livre da fila. Para deslocar o índice em uma posição, é preciso fazer a operação MOD (resto da divisão) de *num_eventos* por 10, que é o número das posições que tem a fila, denominadas por *fila_x* ($x = 1, \dots, 10$). Feito isso, uma instrução tipo CASE insere o valor da variável *eventos* em cada uma das posições da fila.

O passo 2, Figura 4.7, consiste em criar um ponteiro para poder remover elementos da fila. Este ponteiro denominado *indice* aponta para o primeiro elemento da fila que ainda não foi lido e faz parte de outra instrução tipo CASE (inserida em baixo da implementação da fila) que dependendo do seu valor, retira e copia o elemento pertencente a essa posição na variável *evento_a_ler*. Esta última variável armazena os eventos ocorridos no sistema gerando dessa forma o histórico de eventos. A atualização da variável *indice* é realizada por um *Point Link* (Link de Ponto) no software SCADA, que será explicado no passo 4.

No entanto, é necessário se certificar de que a posição apontada pelo *indice* já foi lida, isto é, que o valor contido naquela posição já foi removido. Logo é criada a variável *indice_a_ler* que toma o valor da variável índice. Cabe mencionar que após o código anterior encontra-se o código referente aos supervisores.

```
CASE indice OF
  0 : evento_a_ler := fila_1;
  1 : evento_a_ler := fila_2;
  2 : evento_a_ler := fila_3;
  3 : evento_a_ler := fila_4;
  4 : evento_a_ler := fila_5;
  5 : evento_a_ler := fila_6;
  6 : evento_a_ler := fila_7;
  7 : evento_a_ler := fila_8;
  8 : evento_a_ler := fila_9;
  9 : evento_a_ler := fila_10;
END_CASE;

indice_a_ler := indice;
....
```

Figura 4.7 Passo 2: Criação de ponteiro para retirar elementos da fila

No passo 3, Figura 4.8, a variável *indice_a_ler* junto com a variável *num_eventos* ficam disponíveis no contexto de um *script* gerado no

software SCADA, para atualizar a variável *indice* do passo 2. O *script* testa a condição, se o índice atual da fila (aquele que foi lido) é diferente do índice objetivo (aquele que se deseja ler). O índice objetivo aponta ao primeiro elemento da fila, ou seja, um evento que aconteceu e que ainda não foi removido. Se a expressão for verdadeira, o próximo índice é a posição seguinte àquela anteriormente lida.

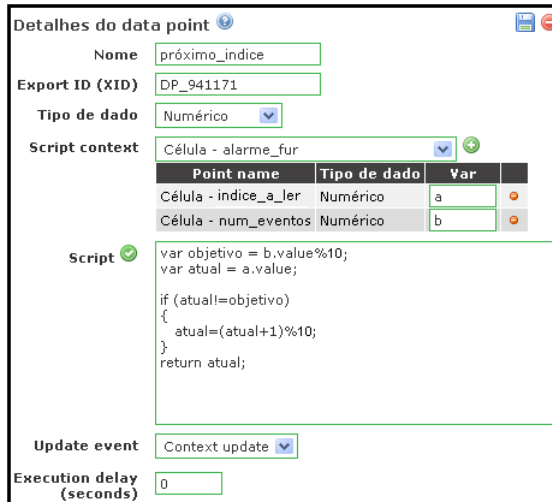


Figura 4.8 Passo 3: Cálculo da posição da fila que deve ser lida

No passo 4, Figura 4.9, é configurado o *Point Link* que faz a escrita ou “ligação” entre a saída do *script* gerado no passo 3 e a variável que se quer atuar, neste caso, a variável *indice*. Assim, esta variável é atualizada e, por conseguinte outro elemento é retirado da fila atualizando a variável *evento_a_ler*. Os *Point links* são utilizados para atualizar o valor de *tags* baseados no valor de outros *tags*. Mantendo os dois *tags* em sincronia de acordo com alguma relação definida pelo usuário. De maneira mais direta, *os point links* podem ser utilizados para ler valores em um sistema (de um ou mais data sources) e logo definir valores em outro sistema (para um ou mais data sources). Os *point links* não iniciam eventos, diminuindo o overhead do sistema e resultando em melhor performance, permitem a definição de scripts, dando mais liberdade ao usuário e a sua interface de usuário está em uma única e simples página, tornando mais fácil sua configuração.

Figura 4.9 Passo 4: Atualização de ponteiro com a nova posição a ser lida

O histórico gerado até aqui, através da variável *evento_a_ler*, mostra números associados a eventos. Porém, é indispensável gerar um histórico onde se correlacione os números com o significado dos eventos. Então, no passo 5 é gerado um *script* no software SCADA com uma instrução tipo CASE de retorno alfanumérico para a variável *evento_a_ler*, como ilustrado na Figura 4.10.

Figura 4.10 Passo 5: Geração de histórico de eventos alfanumérico para a CFM

4.2.4 Geração de alarmes críticos

São chamados de alarmes críticos, aqueles gerados quando uma variável ou condição do processo de produção está fora dos valores previstos transgredindo uma especificação de controle, onde se faz necessária intervenção imediata do operador.

- **Exemplo CFM**

Para exemplificar esta funcionalidade, é implementado na CFM um alarme crítico a partir da variável *unsafe* a qual foi introduzida na seção 4.2.2. Essa variável é ativada quando é executado no modo Manual, um evento cujo sinal de desabilitação tem valor lógico VERDADEIRO. Assim, é notificado o fato ao operador por meio de um alarme chamado de *sistema não seguro* no sinótico da CFM e na tela de alarmes.

Para implementar esse alarme, é modificado o código de eventos controláveis do sistema produto, ao inserir as variáveis *forca equipamento*, *unsafe* e *pausa*, sendo *equipamento* qualquer um dos subsistemas que conformam a planta. Na Figura 4.11, é mostrado como exemplo, um trecho do código onde é implementado o evento controlável *e_afur* do gerador *G₅* que representa a estação de furação da CFM.

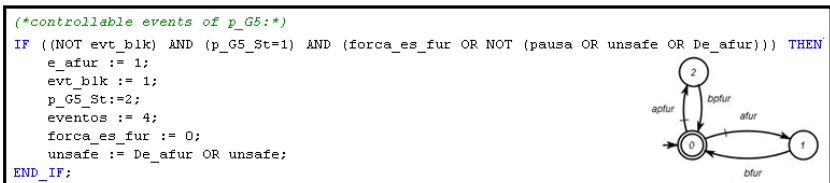


Figura 4.11 Trecho de código do evento controlável *e_afur*

Para passar do estado 0 ao estado 1, é necessário que ocorra o evento *e_afur*, que seu sinal de desabilitação esteja desativado e que a variável *evt_blk* tenha valor lógico zero. No entanto, no modo Manual, podem ser ativadas mais duas variáveis: *forca_es_fur* que atua como um comando para acionar a furadeira e *pausa* que não permite a execução de eventos controláveis quando nesse estado. Agora, se *forca_es_fur* é ativada, *unsafe* toma o valor do sinal de desabilitação de *e_afur*. Caso esse sinal tenha valor lógico 1, *unsafe* será ativada e continuará com esse valor até ser reinicializado o sistema.

4.2.5 Geração de alarmes gerais

São chamados de alarmes gerais, aqueles que fornecem indicação global do estado do sistema, chamam a atenção do operador para uma modificação do estado do processo ou sinalizam um objeto atingido. Estes alarmes devem ser reconhecidos por parte do operador e combinam a informação de sensores com o estado lógico dos modelos dos geradores.

- **Exemplo CFM**

Para exemplificar esta funcionalidade, são apresentados dois alarmes importantes para a CFM, a partir da combinação entre os sinais dos sensores capacitivos da furadeira e soldadeira e os estados dos geradores da estação de furação (gerador G_5) e de soldagem (gerador G_6). Na Figura 4.12, é mostrado o código que implementa esses alarmes. As variáveis s_{fur} , s_{sol} , p_{G5} e p_{G6} são os sinais dos sensores capacitivos da furadeira e soldadeira, os geradores da estação de furação e de soldagem, respectivamente. As duas instruções IF testam a condição se os sinais dos sensores da estação de furação e soldagem estão desativados enquanto os estados 1 ou 2 do gerador G_5 ou do gerador G_6 , os quais representam na sequência, furação, não furação, soldagem e não soldagem, estão ativos. Caso o teste seja positivo, os alarmes são gerados e exibidos na tela do sinótico e de alarmes.

```
IF NOT s_fur AND (p_G5_St=1 OR p_G5_St=2 ) THEN
    alarme_fur :=1;
END_IF;

IF NOT s_sol AND (p_G6_St=1 OR p_G6_St=2) THEN
    alarme_sol :=1;
END_IF;
```

Figura 4.12 Código que implementa alarmes gerais

4.2.6 Gráficos de tendências

Esta funcionalidade, incluída em todo software SCADA, desenha um gráfico de uma variável, contínua ou discreta, em função do tempo. Em sistemas de manufatura, é comum gerar gráficos de tendências de

variáveis relacionadas com o volume de produção diário, semanal, mensal; percentagem de produtos defeituosos; estoque intermediário, etc.

- **Exemplo CFM**

Para a CFM, é implementado um gráfico de tendências que relaciona a contagem de peças que entram no sistema contra aquelas que são aprovadas, reprovadas e retrabalhadas. A partir deste gráfico, Figura 4.13, pode ser determinada a percentagem de peças aprovadas ou fabricadas corretamente, bem como a percentagem de peças reprovadas ou danificadas, ao longo do tempo. A forma como é obtida a contagem de peças é explicada na seção 4.2.9.

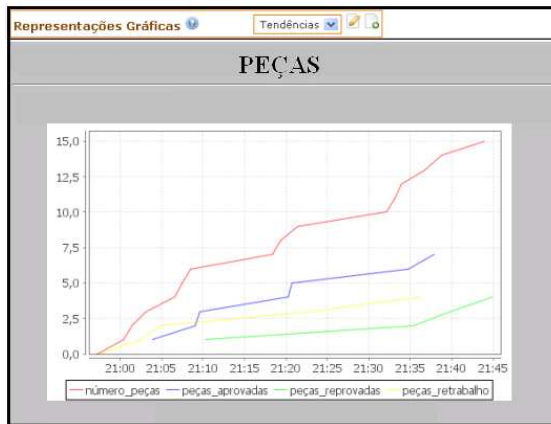


Figura 4.13 Gráfico de tendência para a CFM

4.2.7 Receitas

Conforme foi mencionado no capítulo 2, uma receita é um conjunto de valores pré-definidos que podem ser carregados para um grupo de *tags* a fim de configurar um processo específico.

- **Exemplo CFM**

Devido ao fato de que a CFM só tem um tipo de peça a ser produzida, que deve ser furada e soldada, é implementada uma receita que

faz referência à quantidade de peças a serem fabricadas. Na Figura 4.14 pode ser visto o código gerado no CLP para implementar a receita.

```
(*Receita*)  
  
IF (e_bretok) THEN  
  num_aprovada:= num_aprovada + 1;  
END_IF;  
  
IF (num_aprovada<>0) AND (receita_scada = num_aprovada) THEN  
  pausa := 1;  
  num_aprovada :=0;  
  receita_scada:=0;  
END_IF;
```

Figura 4.14 Código que implementa a receita para a CFM

A sinalização do fim da operação de retirar peça da posição S1e colocar no compartilhamento de peças aprovadas é identificada pela variável *e_bretok*. Por sua vez, a variável *receita_scada*, determina a quantidade de peças a serem fabricadas. Este número é especificado pelo operador na representação gráfica da receita, Figura 4.15. Portanto, quando ocorre *e_bretok* é realizada uma contagem de peças aprovadas através da variável *num_aprovada*. Feito isso, é testada a condição se o valor atribuído à *receita_scada* é igual ao valor armazenado em *num_aprovada*, sendo que esta deve ser diferente de 0. Então, se o teste da afirmação é positivo, suspende-se a geração de eventos controláveis, para garantir que não serão fabricadas mais peças até não ser inserida em *receita_scada* uma nova requisição de peças.

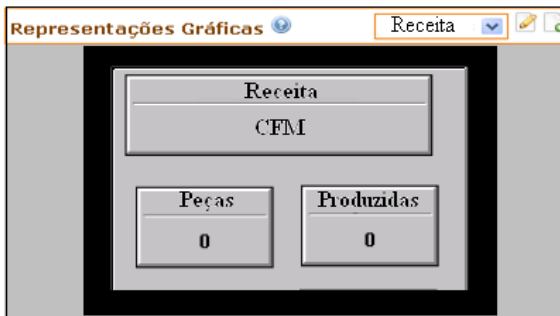


Figura 4.15 Representação gráfica da receita para a CFM

4.2.8 Relatórios

Os relatórios no software SCADA permitem realizar a impressão dos dados históricos, cabeçalhos e alarmes, e ainda dados instantâneos que podem ser agendados automaticamente ou enviados via e-mail. De acordo com a programação do controle supervisório, podem ser configurados relatórios que mostrem a evolução do sistema dinâmico acompanhados dos alarmes gerados por falhas no sistema de controle.

- **Exemplo CFM**

A Figura 4.16 mostra um relatório parcial para a variável *evento_alfanumérico* associada à variável *evento_a_ler*, discutida na seção 4.2.3. Para a CFM foram gerados relatórios importantes dos alarmes da seção 4.2.4 e 4.2.5 e de peças aprovadas, reprovadas e de retrabalho.

scadaBR <small>powered by Serotonin's Mango M2M</small>		RELATORIO HISTÓRICO DE EVENTOS DO SISTEMA	
User access			
	Username	AccessType	
	admin	admin	
	Datapoint	Eventos	
	Evento_Alfanumérico	Todos os eventos	
Histórico		Views	
	Valor	Tempo	Nome
	bretret: Término do transporte de peça de S1 até o	Dez 14 21:34	Data point não utilizado em nenhuma de suas representações gráficas
	aretret: Início do transporte de uma peça de S1 até	Dez 14 21:33	
	b_mes: Término do giro da mesa	Dez 14 21:33	
	a_mes: Início do giro da mesa	Dez 14 21:33	
	b_nok1: Término do teste indicando que a peça dev	Dez 14 21:33	
	ates_1: Início do teste para peças vindas do BE	Dez 14 21:33	
	b_mes: Término do giro da mesa	Dez 14 21:33	
	a_mes: Início do giro da mesa	Dez 14 21:33	
	b_sol: Término da soldagem	Dez 14 21:33	
	a_sol: Início da soldagem	Dez 14 21:32	
	b_mes: Término do giro da mesa	Dez 14 21:32	
	a_mes: Início do giro da mesa	Dez 14 21:32	
	b_fur: Término da furação	Dez 14 21:32	
	a_fur: Início da furação	Dez 14 21:32	
	b_mes: Término do giro da mesa	Dez 14 21:32	
	a_mes: Início do giro da mesa	Dez 14 21:32	
			Eventos
			Id
			Nível de alarme
			Sem eventos para listar

Figura 4.16 Relatório do histórico de eventos para a CFM.

4.2.9 Geração de informação para níveis gerenciais

O sistema SCADA, além de mostrar os dados controlados, pode pré formatá-los para armazenamento em um BD para fins de arquivo

histórico, o que acaba levando a sua integração com outros níveis de gerenciamento fabril como os Sistemas de Gestão Integrado (Enterprise Resource Planning - ERP), sistemas MES (Manufacturing Execution Systems), entre outros. Assim, a geração de informação para níveis gerenciais forma parte essencial de um sistema de produção, permitindo um *feedback* entre as diversas áreas de uma empresa fabril. Esta funcionalidade pode ser implementada através de *scripts* no software SCADA ou de código no CLP auxiliados pela criação ou identificação de variáveis no controle supervisório.

- **Exemplo CFM**

Para exemplificar de uma forma simples, foi gerada a contagem de peças que entram no sistema, peças que são aprovadas, reprovadas e que precisam de retrabalho, concluindo que é uma informação relevante para o sistema gerencial. Para isto, no software SCADA foram criados dois *scripts* (Figura 4.17). O primeiro toma como variável de entrada o estado do gerador G3 (Manipulador Robótico) para determinar que tipo de peça é retirada da mesa giratória para ser depositada num dos três compartimentos: peças aprovadas (estado 1), reprovadas (estado 2) e de retrabalho (estado 3), bem como o número de peças que entram no sistema (estado 4 e 5). O segundo *script* retorna o número de peças em uma hora (podendo ser qualquer faixa de tempo) mediante a função *get()* da linguagem *Ecma-Script*.

Point name	Tipo de dado	Var
Célula - PG3	N Numérico	a

```
if ((a.value == 4) || (a.value == 5))
{
return true;
}
```

Point name	Tipo de dado	Var
Meta_Auxiliar - peças	Binário	p2

```
return p2.past(HOUR).data.get(0).starts
```

Figura 4.17 Scripts para implementar número de peças de entrada

4.3 CONCLUS3O DO CAP3TULO

Neste cap3tulo, foi apresentada uma metodologia que integra o controle supervisi3rio programado em CLP no desenvolvimento de sistemas SCADA para SED. Esta metodologia pode ser implementada em projetos ou re-projetos de automa3o permitindo uma revis3o cont3nua dos resultados obtidos em cada etapa para alcan3ar um funcionamento 3timo do sistema de controle. Conv3m ressaltar, que esta metodologia pode ser desenvolvida em sistemas de manufatura j3 implantados, cuja estrutura de controle seja baseada na ACS, e programada se poss3vel em linguagem de texto estruturado. Sendo assim, o projetista pode seguir esta metodologia a partir da Fase V e voltar 3s fases anteriores se necess3rio, sempre que seja compreendido como 3 feita a s3ntese e programa3o de controle do processo.

No cap3tulo 5, a metodologia de desenvolvimento de sistemas SCADA com controle supervisi3rio 3 aplicada a outra c3lula de manufatura localizada no LAI.

5. APLICAÇÃO DA METODOLOGIA A UMA CÉLULA FLEXÍVEL DE MANUFATURA

Como visto no capítulo 4, os sistemas SCADA exploram as características do controle supervisão programado em CLP através de um desenvolvimento integrado que segue várias fases. Neste capítulo é apresentado um estudo de caso que aplica a metodologia proposta no capítulo 4 para controlar e supervisionar uma célula flexível de manufatura didática (CFM) da marca RHINO, a qual simula um processo real de fabricação e montagem. O produto final é composto por 4 tipos de peças: uma base, um disco, um anel e um cilindro. A partir desta metodologia, são realizadas as fases de síntese de controle supervisão, implementação de funcionalidades do sistema SCADA, entre outras para a CFM.

5.1 FASE I: PROJETO INFORMACIONAL

A célula flexível de manufatura didática fabricada pela empresa norte-americana *RHINO Robotics Ltd.* e localizada no Laboratório de Automação Industrial (LAI) do Departamento de Automação e Sistemas (DAS) da Universidade Federal de Santa Catarina, está composta pelos seguintes equipamentos (ver Figura 5.1):

- 4 Alimentadores, cada um com um tipo específico de peça: (1) base, (2) anel, (3) disco e (4) cilindro.
- 3 Estações de trabalho: (5) centro de usinagem, (6) sistema de visão e (7) torno.
- 1 Robô articulado XR4 de 6 graus de liberdade (6) mais uma mesa X-Y.
- 1 Robô articulado SCARA com 5 graus de liberdade (9).
- 1 Esteira de alimentação (10).
- 1 Esteira de saída (11).
- 1 Mesa de montagem com 2 posições (12.a e 12.b).
- 1 *Buffer* de descarte de peça (16).

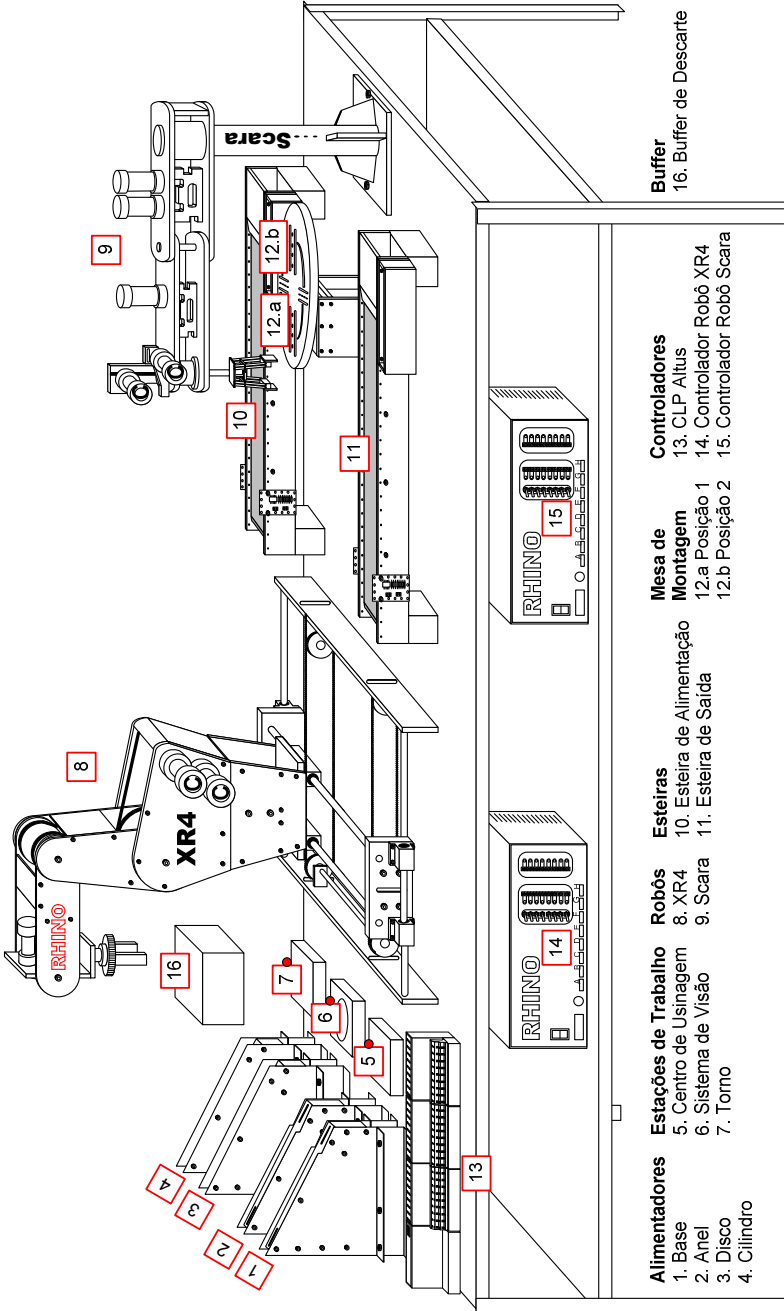


Figura 5.1 CFM Sistema RHINO

A CFM simula um processo real de fabricação e montagem de um produto que está composto pelo conjunto de 4 tipos de peças, que são alimentadas no sistema através dos alimentadores: base, anel, disco e cilindro. O processo de fabricação está representado por 3 operações: centro usinagem, torno CNC e sistema de visão (detecção de falhas). As peças aprovadas são encaminhadas para a esteira de alimentação e peças rejeitadas são depositadas no *buffer* de descarte. Por sua vez, o processo de montagem é realizado pelo robô SCARA, que deve montar as peças finais na ordem correta, segundo a sequência das peças existentes na esteira. A sequência de montagem de um conjunto de peças é descrita a seguir.

Inicialmente o robô XR4 retira a peça do tipo base do seu respectivo alimentador de peças, e a deposita no centro de usinagem. Ao final da usinagem da base, o robô a retira do centro e a deposita no sistema de visão, para a inspeção da peça. Caso a base seja aprovada, então o robô retira a peça disco do seu respectivo alimentador, e a coloca no chanfro existente na base, para então retirá-la e depositá-la na esteira de alimentação. Caso a base seja rejeitada, então o robô a deposita no *buffer* de descarte.

Posteriormente, a peça do tipo anel é retirada pelo robô XR4 do seu respectivo alimentador, e a deposita no torno CNC. Ao final do torneamento, o robô a retira e a deposita no sistema de visão. Caso a peça seja aprovada, então o robô retira a peça do sistema de visão e a deposita na esteira de alimentação, e caso a peça seja reprovada, então ela é depositada no *buffer* de descarte. Este mesmo procedimento é realizado para a peça do tipo cilindro.

A esteira deve avançar as peças na direção da posição de retirada de peças pelo robô SCARA. Nessa posição, o robô SCARA retira da esteira de alimentação a peça base, com o disco já previamente fixado, e a deposita, numa das duas posições livres da mesa de montagem. Na sequência, para cada base, o robô SCARA deverá colocar 1 unidade da peça do tipo anel e outra do tipo cilindro. Ao final de um conjunto montado, o robô SCARA retira tal conjunto da mesa de montagem e o deposita na esteira de saída. O conjunto pronto pode ser visualizado na Figura 5.2.



Figura 5.2 Conjunto final

O funcionamento das estações de trabalho é simulado através de uma rotina de temporização. Um *led* é ativado indicando fim de operação. O centro de usinagem é usado para processar a peça do tipo base, iniciando seu funcionamento ao chegar uma peça; o sistema de visão é utilizado para inspecionar as peças tipo base, cilindro e anel, sendo que as peças rejeitadas são encaminhadas para o *buffer* de descarte e as peças aprovadas, para a esteira de alimentação; já o torno CNC é usado para processar as peças tipo anel e cilindro e inicia seu funcionamento ao chegar uma peça. A operação de teste do sistema de visão é simulada através de uma função RANDOM.

Sempre que uma peça chega aos alimentadores, é ativado um sensor de toque. Para as peças do tipo disco se considera que sempre estão disponíveis no sistema.

Cada esteira tem um sensor de presença, porém apenas o da esteira de alimentação é utilizado. Um motor DC com *encoder* é responsável pela velocidade de cada esteira. A esteira de alimentação possui 3 posições: posição de depósito de peça pelo robô XR4, posição intermediária e posição de retirada de peça pelo robô SCARA. A esteira de saída Avança toda vez que é depositado um conjunto pronto.

O sistema RHINO é controlado através de um CLP da marca Altus da série Ponto, Figura 5.1 (13). O barramento do CLP Altus PO3147 possui 3 módulos: PO1010, PO2020 e PO2022. O módulo PO1010 possui 32 pontos de entrada digital para tensão de 24Vdc. O módulo PO2020 possui 16 pontos de saída digital transistorizados isolados e com alimentação comum. Já o módulo PO2022 possui 16 pontos de saída digital com contatos secos, ideal para o acionamento de cargas em corrente contínua ou alternada e para inserção no intertravamento de circuitos lógicos. Além disso, o CLP Altus possui três canais ou portas seriais, sendo COM1 o principal e o COM2 e COM3 os auxiliares e suporta o protocolo proprietário ALNET I disponível em todos os canais seriais e o protocolo MODBUS RTU, disponível apenas nos canais auxiliares. Maiores informações sobre o hardware e a programação do CLP podem ser obtidas diretamente no site da empresa Altus, nas seções de documentação técnica e tutoriais.

Os Robôs XR4 e SCARA são robôs didáticos desenvolvidos com as mesmas características de um robô industrial, porém sua estrutura permite a observação das engrenagens e mecanismos de movimentação. Algoritmos de controle PID permitem controlar a velocidade completa de todos os servo motores, 6 para o robô XR4 e 5 para o robô SCARA. O robô XR4 locomove-se sobre uma mesa X-Y acionada por 2 servo motores. O controle de cada robô é feito por um controlador dedicado

Mark IV, Figura 5.1 (14) e (15), e sua programação é feita utilizando-se um programa com instruções padronizadas na linguagem de programação do equipamento chamada RoboTalk, o qual é instalado num computador de mesa. Cada controlador Mark IV possui 8 entradas e saídas digitais em resolução de 8 bits e um *teach pendant* para configuração manual.

A arquitetura do sistema RHINO é ilustrada na Figura 5.3. Num primeiro nível, encontra-se o servidor SCADA ligado ao cliente que tem a interface homem máquina (IHM). O servidor está conectado ao CLP Altus através de uma comunicação serial RS232 com o protocolo Modbus RTU e o CLP está conectado ao resto de subsistemas. Os *leds* das estações de trabalho e os motores DC das esteiras são conectados às saídas do módulo PO2022 do CLP. Os sensores dos alimentadores são conectados às entradas do módulo PO1010. Por sua vez, os controladores dos robôs XR4 e SCARA são conectados aos computadores de mesa onde está rodando o programa na linguagem RoboTalk, mediante uma comunicação RS232C. As saídas dos módulos PO2020 e PO2022 são conectadas às entradas dos controladores do robô XR4 e SCARA, respectivamente, e as saídas dos controladores são conectadas às entradas do módulo PO1010.

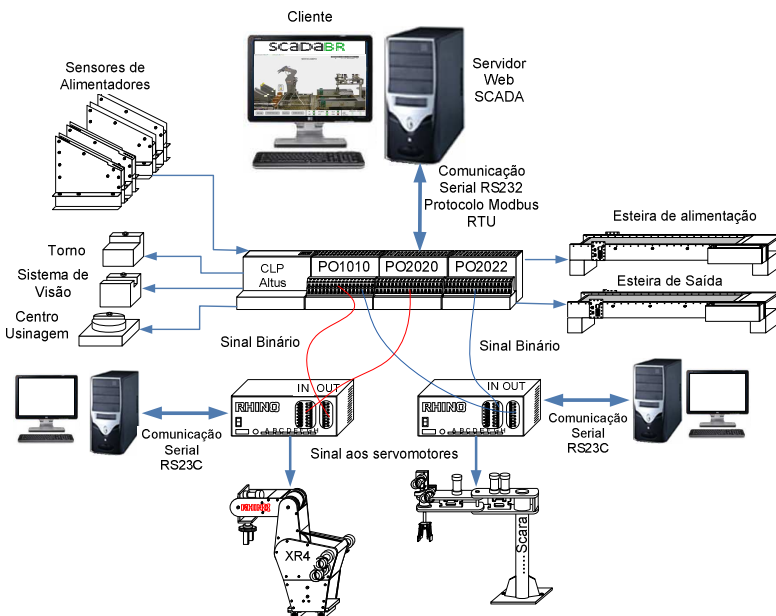


Figura 5.3 Arquitetura Sistema RHINO

As ferramentas utilizadas na aplicação desta metodologia são: IDES e TCT para realizar a construção de geradores tanto da planta quanto das especificações, assim como o cálculo e redução de supervisores; Supremica para realizar a emulação dos geradores obtidos na fase II, Ides2ST e MasterTool para a implementação da fase IV e o software ScadaBR para a implementação das fases V e VI.

5.2 FASE II: SÍNTESE DE CONTROLE SUPERVISÓRIO

Visando a fabricação de conjuntos de peças de forma eficiente e flexível, alguns problemas devem ser evitados, como:

- Realizar a montagem de mais de 2 conjuntos de peças simultaneamente.
- Processar no torno um anel e um cilindro ao mesmo tempo.
- Processar no sistema de visão mais de uma peça.
- Efetuar as operações de usinagem, torneamento e teste se não houver peça esperando pela realização das tarefas citadas.
- Pegar peças dos alimentadores por parte do robô XR4 se não houver pelo menos uma peça disponível.
- Overflow e underflow da esteira de alimentação.
- Overflow da esteira de saída.
- Permitir desordem na sequência de peças que entram na esteira de alimentação para depois ser montadas.
- Realizar a montagem dos conjuntos de forma errada. Um conjunto está formado por 1 unidade de cada tipo de peça e as peças cilindro e anel devem ser montadas na mesa de montagem sobre uma peça do tipo base.

5.2.1 Etapa 2.1 Modelagem dos subsistemas da CFM

Na tabela 5.1 são apresentados os eventos utilizados na modelagem dos geradores dos subsistemas e o significado físico dos mesmos.

Tabela 5.1 Eventos do Sistema RHINO

Evento	Descrição
Alimentadores	
Chega_A	Chega peça anel no seu alimentador
Chega_B	Chega peça base no seu alimentador

Chega_C	Chega peça cilindro no seu alimentador
Chega_D	Chega peça disco no seu alimentador
Torno CNC	
I_tor_A	Início de torneamento de peça anel
F_tor_A	Sinaliza o fim desta operação
I_tor_C	Início de torneamento de peça cilindro
F_tor_C	Sinaliza o fim desta operação
Sistema de Visão	
I_tes_A	Início de teste de peça anel
F_tes_A_ok	Sinaliza a aprovação no teste
F_tes_A_nok	Sinaliza a aprovação no teste
I_tes_B	Início de teste de peça base
F_tes_B_ok	Sinaliza a aprovação no teste
F_tes_B_nok	Sinaliza a aprovação no teste
I_tes_C	Início de teste de peça cilindro
F_tes_C_ok	Sinaliza a aprovação no teste
F_tes_C_nok	Sinaliza a aprovação no teste
Centro Usinagem	
I_us_B	Início de usinagem de base
F_us_B	Sinaliza o fim desta operação
Robô XR4	
I_col_B_us	Início do transporte de peça base do seu alimentador até o centro de usinagem
F_col_B_us	Sinaliza o fim desta operação
I_col_B_sv	Início do transporte de peça base do centro de usinagem até o sistema de visão
F_col_B_sv	Sinaliza o fim desta operação
I_col_D_bsv	Início do transporte de peça disco do seu alimentador até chanfro de peça base no sistema de visão
F_col_D_bsv	Sinaliza o fim desta operação
I_col_BD_ea	Início do transporte do conjunto base + disco do sistema de visão até a esteira de alimentação
F_col_BD_ea	Sinaliza o fim desta operação
I_col_B_nok	Início do transporte de peça base do sistema de visão até o <i>buffer</i> de descarte
F_col_B_nok	Sinaliza o fim desta operação
I_col_A_tenc	Início do transporte de peça anel do seu alimentador até o torno CNC
F_col_A_tenc	Sinaliza o fim desta operação
I_col_A_sv	Início do transporte de peça anel do torno CNC até o sistema de visão
F_col_A_sv	Sinaliza o fim desta operação

I_col_A_ea	Início do transporte de peça anel do sistema de visão até a esteira de alimentação
F_col_A_ea	Sinaliza o fim desta operação
I_col_A_nok	Início do transporte de peça anel do sistema de visão até o <i>buffer</i> de descarte
F_col_A_nok	Sinaliza o fim desta operação
I_col_C_tcnc	Início do transporte de peça cilindro do seu alimentador até o torno CNC
F_col_C_tcnc	Sinaliza o fim desta operação
I_col_C_sv	Início do transporte de peça cilindro do torno CNC até o sistema de visão
F_col_C_sv	Sinaliza o fim desta operação
I_col_C_ea	Início do transporte de peça cilindro do sistema de visão até a esteira de alimentação
F_col_C_ea	Sinaliza o fim desta operação
I_col_C_nok	Início do transporte de peça cilindro do sistema de visão até o <i>buffer</i> de descarte
F_col_C_nok	Sinaliza o fim desta operação
Esteira de alimentação	
I_ea	Início do avanço da esteira de alimentação
F_ea	Sinaliza o fim desta operação
Esteira de saída	
I_es	Início do avanço da esteira de saída
F_es	Sinaliza o fim desta operação
Aduana	
I_Aduana_A	Início de <i>inspeção</i> de peça anel, segunda posição da esteira de alimentação
F_Aduana_A	Sinaliza o fim desta operação
I_Aduana_BD	Início de <i>inspeção</i> de peça base + disco, segunda posição da esteira de alimentação
F_Aduana_BD	Sinaliza o fim desta operação
I_Aduana_C	Início de <i>inspeção</i> de peça cilindro, segunda posição da esteira de alimentação
F_Aduana_C	Sinaliza o fim desta operação
Robô SCARA	
I_col_BD_mm1	Início do transporte de conjunto base + disco da esteira de alimentação até a posição 1 da mesa de montagem
F_col_BD_mm1	Sinaliza o fim desta operação
I_col_BD_mm2	Início do transporte de conjunto base + disco da esteira de alimentação até a posição 2 da mesa de montagem

F_col_BD_mm2	Sinaliza o fim desta operação
I_col_A_mm1	Início do transporte de peça anel da esteira de alimentação até a posição 1 da mesa de montagem
F_col_A_mm1	Sinaliza o fim desta operação
I_col_A_mm2	Início do transporte de peça anel da esteira de alimentação até a posição 2 da mesa de montagem
F_col_A_mm2	Sinaliza o fim desta operação
I_col_C_mm1	Início do transporte de peça cilindro da esteira de alimentação até a posição 1 da mesa de montagem
F_col_C_mm1	Sinaliza o fim desta operação
I_col_C_mm2	Início do transporte de peça cilindro da esteira de alimentação até a posição 2 da mesa de montagem
F_col_C_mm2	Sinaliza o fim desta operação
I_col_Pronta1_es	Início do transporte do conjunto final montado da posição 1 da mesa de montagem até a esteira de saída
F_col_Pronta1_es	Sinaliza o fim desta operação
I_col_Pronta2_es	Início do transporte do conjunto final montado da posição 2 da mesa de montagem até a esteira de saída
F_col_Pronta2_es	Sinaliza o fim desta operação

Os alimentadores de peças: anel, base, cilindro e disco possuem sensores cujos modelos – G_1 , G_2 , G_3 , e G_4 – são vistos na Figura 5.4. Os sensores sinalizam a chegada de peças através dos eventos *chega_a*, *chega_b*, *chega_c* e *chega_d* respectivamente. Os eventos são modelados como não controláveis e podem ocorrer a qualquer instante no sistema.

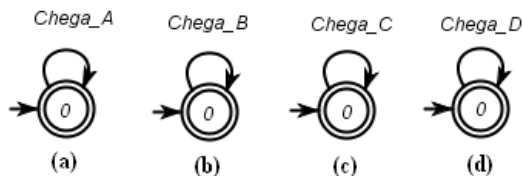


Figura 5.4 Geradores dos alimentadores (a) G_1 , (b) G_2 , (c) G_3 e (d) G_4

O gerador do torno – G_5 –, ilustrado pela Figura 5.5 (a), inicia a operação de torneamento de um anel ou um cilindro, a partir do comando I_{tor_A} e I_{tor_C} respectivamente e ao final da operação envia o sinal F_{tor_a} e F_{tor_C} .

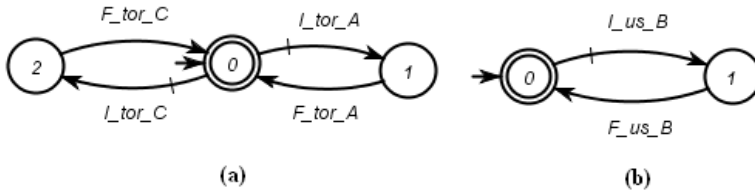


Figura 5.5 Geradores do torno e usinagem (a) G_5 e (b) G_7

O sistema de visão inspeciona peças para encontrar defeitos que não permitam a montagem do conjunto final. Para isto, é modelado o gerador – G_6 –, visto na Figura 5.6, o qual realiza, logicamente, o teste das peças: anel, base e cilindro, através dos comandos I_{tes_A} , I_{tes_B} e I_{tes_C} respectivamente e responde com $F_{tes_A_ok}$, $F_{tes_B_ok}$ e $F_{tes_C_ok}$ caso as peças tenham sido aprovadas ou $F_{tes_A_nok}$, $F_{tes_B_nok}$ e $F_{tes_C_nok}$ se rejeitadas. O centro de usinagem – G_7 – ilustrado na Figura 5.5 (b), inicia sua operação mediante o comando I_{us_B} e a finaliza com o sinal F_{us_B} .

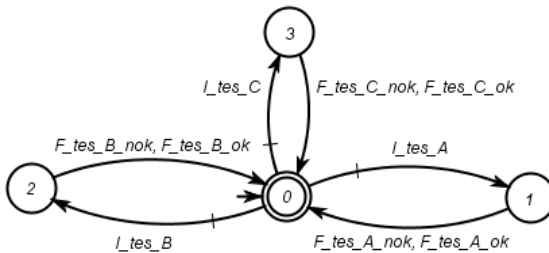


Figura 5.6 Gerador do teste G_6

O robô XR4, representado pelo gerador – G_8 – na Figura 5.7, executa treze ações diferentes. Ele pode pegar um anel ou um cilindro iniciados por $I_{col_A_tcnc}$ e $I_{col_C_tcnc}$ respectivamente, e colocá-los no torno; pode pegar também uma base, iniciado por $I_{col_B_us}$ e colocá-la no centro de usinagem e um disco, iniciado por $I_{col_D_bsv}$ e colocá-lo no chanfro existente na base que está no sistema de visão, após esta última ser aprovada. Ele também retira peças do centro de usinagem e torno uma vez terminadas as operações de usinagem e torneamento respectivamente, e as coloca no sistema de visão, através dos comandos $I_{col_B_sv}$, $I_{col_A_sv}$ e $I_{col_C_sv}$. Além disso, ele pode retirar peças aprovadas do sistema de visão, e depositá-las na esteira de alimentação por meio dos comandos $I_{col_A_ea}$, $I_{col_BD_ea}$ e

$I_{col_C_ea}$ ou peças reprovadas e colocá-las no *buffer* de descarte, através dos comandos $I_{col_A_nok}$, $I_{col_B_nok}$ e $I_{col_C_nok}$.

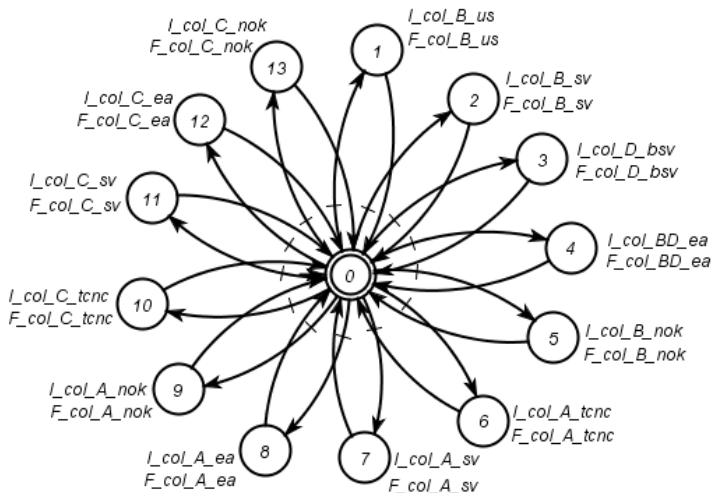


Figura 5.7 Gerador do robô XR4 G_8

O gerador da esteira de alimentação – G_9 –, ilustrado pela Figura 5.8, inicia o avanço a partir do comando I_{ea} e ao final da operação envia o sinal F_{ea} .

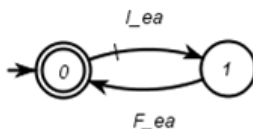


Figura 5.8 Gerador da esteira de alimentação G_9

Visando gerar um histórico de eventos da passagem de peças pela esteira de alimentação, é modelado o gerador – G_{10} – chamado Aduana, ilustrado na Figura 5.9. Este gerador não modela um componente físico do sistema, propriamente, sendo um artifício criado para representar a passagem de um anel, uma base com disco ou um cilindro, da posição 1 para a posição 2 da esteira de alimentação, através dos eventos $I_{aduanas}$, $I_{aduanas_BD}$ e $I_{aduanas_C}$ respectivamente. Os eventos finais de inspeção $F_{aduanas}$, $F_{aduanas_B}$ e $F_{aduanas_C}$ que atuam como sensores, determinam a peça que ocupa a segunda posição da

esteira de alimentação. A introdução destes eventos facilita a modelagem das especificações apresentadas na seção 5.2. Alternativamente, outras abordagens podem ser adotadas para isso, como o uso dos distinguidores (BOUZON et al. 2009)

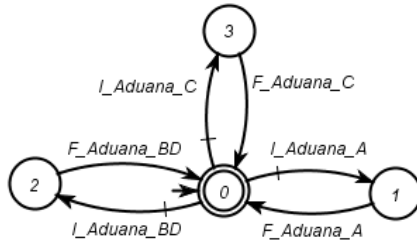


Figura 5.9 Gerador da aduana G_{10}

O robô SCARA – G_{11} –, ilustrado na Figura 5.10, executa oito ações referentes à montagem de dois conjuntos finais para duas posições da mesa de montagem. Ele primeiro retira da esteira de alimentação uma base mais o disco anteriormente colocado nela, através dos comandos $I_col_BD_mm1$, $I_col_BD_mm2$, para depois ser depositado na posição 1 ou 2. Feito isso, ele pode retirar tanto um anel quanto um cilindro da esteira de alimentação pelos comandos $I_col_A_mm1$, $I_col_A_mm2$, $I_col_C_mm1$ e $I_col_C_mm2$ e montá-los sobre a base na posição 1 ou 2. Quando os conjuntos estiverem prontos, ele pode colocá-los sobre a esteira de saída a partir dos comandos $I_col_Pronta1_es$ e $I_col_Pronta2_es$, para os conjuntos vindos da posição 1 ou 2 respectivamente.

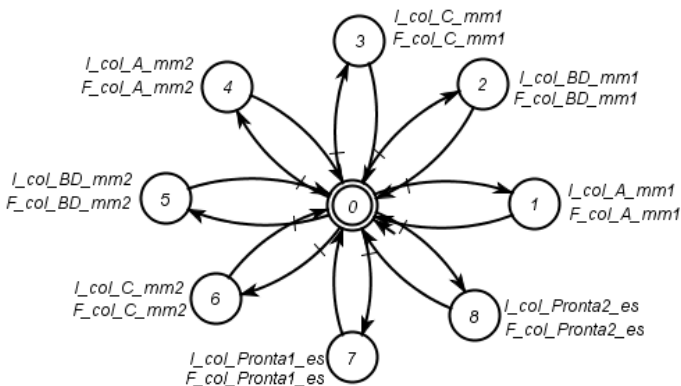


Figura 5.10 Gerador do robô SCARA G_{11}

O gerador da esteira de saída – G_{12} –, visto na Figura 5.11, inicia seu avanço com um conjunto pronto, a partir do comando I_es e quando finaliza sua operação envia o sinal F_es .

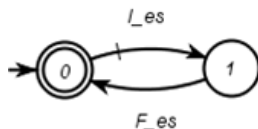


Figura 5.11 Gerador da esteira de saída G_{12}

5.2.2 Etapa 2.2 Modelagem das especificações

As especificações focam na resolução dos problemas de interação entre os subsistemas mencionados na seção 5.2.

As especificações E_{1a} , E_{1b} e E_{1c} , visualizadas na Figura 5.12(a), (b) e (c) respectivamente, objetivam a coordenação entre os alimentadores de anéis, bases e cilindros com o robô XR4. No estado 0 dos geradores, não há peças nos alimentadores, sendo proibido que o robô XR4 inicie o movimento de pegar um anel, uma base e um cilindro de seus respectivos alimentadores – $I_col_A_tcnc$, $I_col_B_us$ e $I_col_C_tcnc$. Estes eventos somente são habilitados no estado 1, quando é certa a existência de pelo menos um anel, uma base e um cilindro em seus alimentadores – após ocorrer os eventos $Chega_A$, $Chega_B$ e $Chega_C$. Os auto-laços em E_{1a} , E_{1b} , E_{1c} , permitem que estes últimos eventos possam ocorrer mais de uma vez. Já a especificação E_{1d} , ilustrada na Figura 5.12 (d), não permite no estado 0 que o robô XR4 retire um disco para colocar sobre uma base no sistema de visão – $I_col_D_bsv$, até não ter certeza da existência de pelo menos um disco no seu alimentador através do evento $Chega_D$.

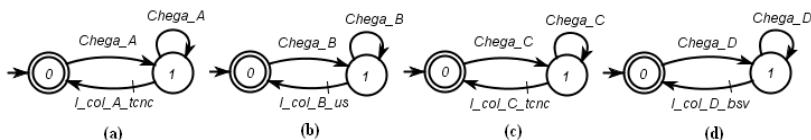


Figura 5.12 Especificação de coordenação dos alimentadores com o robô XR4 (a) E_{1a} , (b) E_{1b} , (c) E_{1c} e (d) E_{1d}

Visando evitar a *underflow* e *overflow* das estações de trabalho: torno, sistema de visão e centro de usinagem, foram modeladas as espe-

cificações E_{2a} , E_{2b} e E_{2c} , as quais desabilitam as operações de retirada de peças se elas não foram depositadas anteriormente e desabilitam um novo depósito até que as mesmas sejam retiradas. O gerador da Figura 5.13 (a) indica que quando há peças no torno – estado 1–, pode ocorrer a retirada de anéis o cilindros – $I_{col_A_sv}$ e $I_{col_C_sv}$ – e quando o torno esta vazio – estado 0 – pode acontecer o depósito das mesmas – $F_{col_A_tcnc}$ e $F_{col_C_tcnc}$. O estado 1 do gerador da Figura 5.13 (b) representa o sistema de visão cheio, ou seja, com um anel, uma base ou um cilindro, onde é permitido a ocorrência do evento $F_{col_D_bsv}$ mais de uma vez pela restrição imposta na seção 5.2, onde a peça disco só pode entrar no sistema de visão para ser colocada sobre a base, quando esta última é aprovada. A partir deste estado, o sistema de visão pode ficar vazio por ações do robô XR4, referente ao depósito de peças na esteira de alimentação ou no *buffer* de descarte.

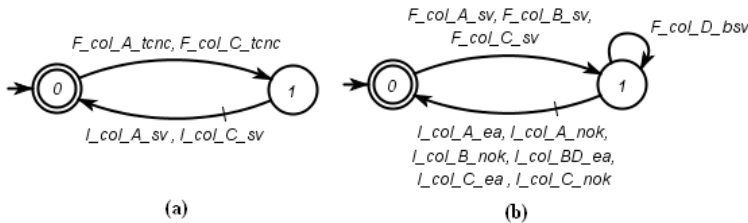


Figura 5.13 Especificação do *buffer* do torno (a) E_{2a} e sistema de visão (b) E_{2b}

A especificação E_{2c} , visualizada na Figura 5.14, indica que somente pode ser retirada uma base – $I_{col_B_sv}$ – se essa está no centro de usinagem – estado 1. Quando esta estação está vazia – estado 0 – pode ser depositada uma base através do evento $F_{col_B_us}$.

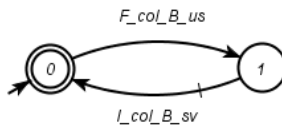


Figura 5.14 Especificação do *buffer* do centro de usinagem E_{2c}

As especificações E_{3a} , E_{3b} e E_{3c} tratam a coordenação do robô XR4 com o fim das operações de torneamento e usinagem. Os geradores da Figura 5.15 (a), (c) e (b) indicam que as operações de torneamento e usinagem – I_{tor_A} , I_{tor_C} e I_{us_B} – somente podem começar quando é colocado um anel ou um cilindro no torno ou uma base no centro de usinagem – estado 1.

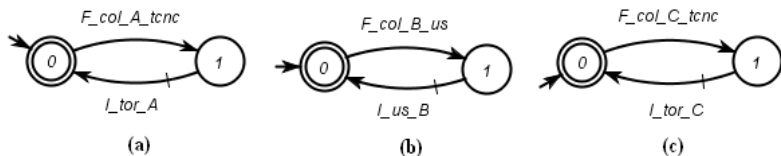


Figura 5.15 Especificações de coordenação do robô XR4 com o torno (a) E_{3a} e (c) E_{3c} e centro de usinagem (b) E_{3b}

As especificações E_{4a} , E_{4b} e E_{4c} respondem pela coordenação entre as estações de torno e usinagem e o robô XR4. Os geradores da Figura 5.16 (a), (c) e (b) indicam que a retirada de peças do torno e do centro de usinagem – $I_{col_A_sv}$, $I_{col_C_sv}$ e $I_{col_B_sv}$ – somente podem começar quando as peças tenham sido manufaturadas – estado 1.

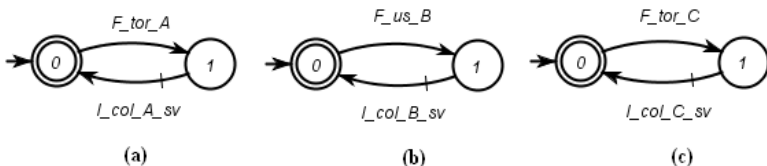


Figura 5.16 Especificações de coordenação do torno e centro de usinagem com o robô XR4 (a) E_{4a} , (b) E_{4b} e (c) E_{4c}

A coordenação do robô XR4 com o sistema de visão é modelada através dos geradores E_{5a} , E_{5b} e E_{5c} , ilustrados na Figura 5.17 (a), (b) e (c). Uma vez depositadas as peças anel, base ou cilindro pelo robô XR4 no sistema de visão – $F_{col_A_sv}$, $F_{col_B_sv}$ e $F_{col_C_sv}$ –, – estado 1 –, pode ser iniciado o teste das peças – I_{tes_A} , I_{tes_B} e I_{tes_C} –, respectivamente.

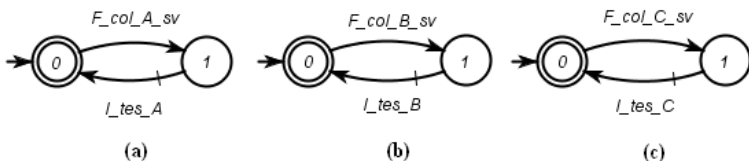


Figura 5.17 Especificações de coordenação do robô XR4 com o sistema de visão (a) E_{5a} , (b) E_{5b} e (c) E_{5c}

Os geradores da Figura 5.18 (a), (b), (c) e (d) são responsáveis da coordenação entre a operação do sistema de visão e o robô XR4. As especificações E_{6a} e E_{6c} desabilitam os eventos de retirada de anéis e cilindros

do sistema de visão, se elas não foram previamente aprovadas ou reprovadas e desabilitam um novo teste até que as mesmas sejam testadas. A especificação E_{6b} desabilita a retirada de uma base do sistema de visão em direção ao *buffer* de descarte se ela não foi testada como reprovada e desabilita o transporte de uma base mais o disco à esteira de alimentação se antes não foi colocado o disco sobre a base – evento $F_{col_D_bsv}$. Já na especificação E_{6d} , não pode ser colocado o disco sobre a base sem que esta última seja aprovada.

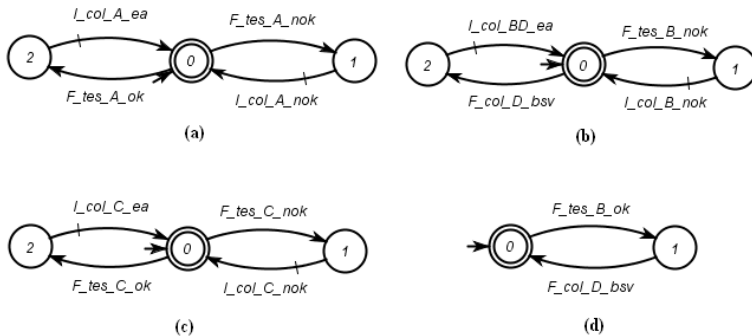


Figura 5.18 Especificações de coordenação do sistema de visão com o robô XR4 (a) E_{6a} , (b) E_{6b} , (c) E_{6c} e (d) E_{6d}

A coordenação dos robôs - XR4 e SCARA - e das 3 posições da esteira de alimentação em relação ao fluxo de peças é subdividida em 2 especificações, de modo que cada uma delas foca na interação entre duas posições vizinhas da esteira. A especificação E_{7a} responde pelo robô XR4 e a aduana, enquanto que a especificação E_{7b} pela aduana e o robô SCARA. Um modelo genérico para os geradores E_{7a} e E_{7b} é mostrado na Figura 5.19. Este modelo segue o mesmo raciocínio adotado para modelar as especificações de coordenação E_{3a} , E_{3b} , E_{3c} e E_{3d} da CFM (YURI, 2010) apresentado no capítulo 3 deste trabalho. Os estados destes geradores são definidos por um par de letras que indicam a presença de um tipo de peça em alguma posição da esteira de alimentação esperando pela execução de alguma operação. Cabe mencionar que as posições da esteira de alimentação estão associadas a subsistemas. A primeira posição está associada ao robô XR4, a segunda à Aduana e a terceira ou última posição está associada ao robô SCARA. O número 0 indica que não há peça em alguma posição da esteira e as letras A, B e C indicam que há um anel, uma base com disco ou um cilindro esperando pela execução de uma tarefa. Os índices dos estados representam respec-

tivamente para a especificação E_{7a} , a primeira e segunda posição e para a especificação E_{7b} , a segunda e terceira posição da esteira de alimentação.

Os eventos $F_{\langle ação_A \rangle}$, $F_{\langle ação_BD \rangle}$ e $F_{\langle ação_C \rangle}$ correspondem aos sinais de fim de operação do subsistema anterior sobre um anel, uma base com disco e um cilindro. Já os eventos $I_{\langle prox_ação_A \rangle}$, $I_{\langle prox_ação_BD \rangle}$ e $I_{\langle prox_ação_C \rangle}$ representam o início das operações sobre um anel, uma base com disco e um cilindro pelo subsistema seguinte ao que sinalizou os eventos $F_{\langle ação_A \rangle}$, $F_{\langle ação_BD \rangle}$ e $F_{\langle ação_C \rangle}$.

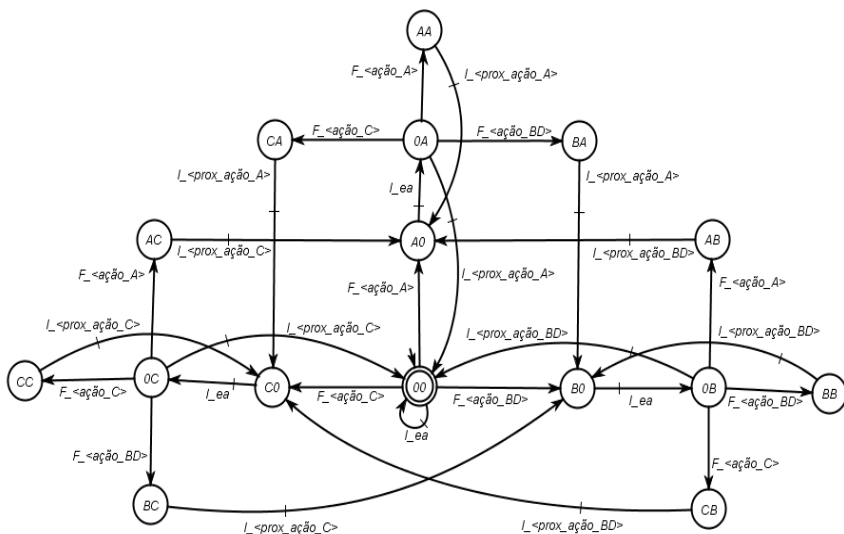


Figura 5.19 Especificações genéricas para a coordenação do robô XR4 com a esteira de alimentação E_{7a} e com a aduana E_{7b}

Suponha que a Figura 5.19 é o gerador que representa a especificação E_{7a} que coordena o robô XR4 e a Aduana: primeira e segunda posição da esteira de alimentação. Para este gerador, os eventos $F_{\langle ação_A \rangle}$, $F_{\langle ação_BD \rangle}$, $F_{\langle ação_C \rangle}$, $I_{\langle prox_ação_A \rangle}$, $I_{\langle prox_ação_BD \rangle}$ e $I_{\langle prox_ação_C \rangle}$ são respectivamente: $F_{col_A_ea}$, $F_{col_BD_ea}$ e $F_{col_C_ea}$, I_{Aduana_A} , I_{Aduana_BD} e I_{Aduana_C} . A Figura 5.20 ilustra uma sequência de eventos a qual permite mostrar a evolução da especificação E_{7a} , supondo inicialmente que a esteira de alimentação esteja vazia. No momento do robô XR4 depositar um anel na primeira posição, o gerador evolui ao estado A0,

Figura 5.20 (a), desabilitando o evento $F_{col_A_ea}$ para evitar o *overflow* de peças nessa posição. Posteriormente é habilitado o avanço da esteira de alimentação para a segunda posição por meio do evento I_{ea} , fazendo evoluir o gerador ao estado 0A, mostrado fisicamente na Figura 5.20 (b). Neste estado pode ser determinada a existência de um anel na segunda posição - I_{Aduana_A} - ou a chegada de uma base com disco - I_{Aduana_BD} - ou um cilindro - I_{Aduana_C} - na primeira posição da esteira de alimentação. Ainda neste estado é proibida a ativação do comando I_{ea} e em todos os estados onde o segundo índice seja diferente de 0, já que significa que existe uma peça sobre a qual não foi realizada a operação esperada. Neste caso, seja um anel, Figura 5.20 (c), uma base com disco ou um cilindro o que chegar à primeira posição, o comando I_{ea} não poderá ser habilitado porquanto ainda não foi inspecionada a peça anel.

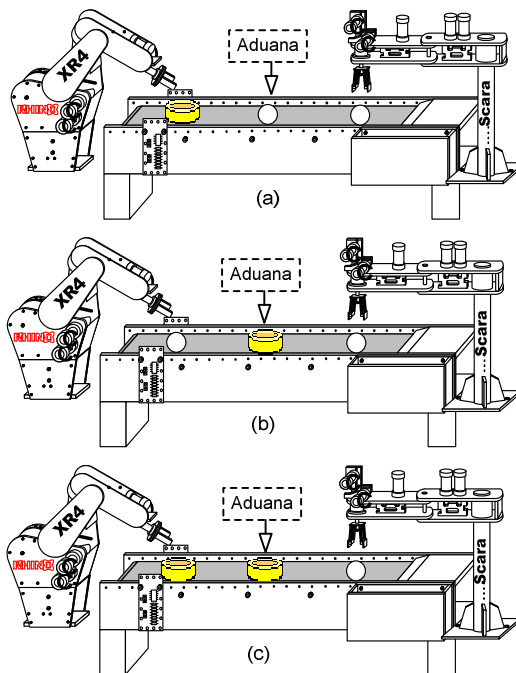


Figura 5.20 Sequência de operações relacionada à especificação E_{7a}

Para a especificação E_{7b} , os eventos $F_{<ação_A>}$, $F_{<ação_BD>}$, $F_{<ação_C>}$ são respectivamente, F_{Aduana_A} ,

F_Aduana_BD e F_Aduana_C . Em virtude dos movimentos do robô SCARA associados à última posição da esteira de alimentação serem dirigidos à montagem de peças em duas posições diferentes da mesa de montagem, o evento $I_{\langle prox_açãõ_A \rangle}$ equivale a $I_{col_A_mm1}$ ou $I_{col_A_mm2}$; $I_{\langle prox_açãõ_BD \rangle}$ a $I_{col_BD_mm1}$ ou $I_{col_BD_mm2}$ e por fim, $I_{\langle prox_açãõ_C \rangle}$ a $I_{col_C_mm1}$ ou $I_{col_C_mm2}$. Estas duas especificações juntas asseguram que não haverá *overflow* na primeira posição nem *underflow* na última posição da esteira de alimentação; esta última não avançará antes de as peças: anel, base com disco ou cilindro serem depositadas pelo robô XR4 na primeira posição, “inspeccionadas” pela aduana e retiradas pelo robô SCARA na última posição; não ocorrerá a “inspeção” da aduana sem ter peça na segunda posição.

A especificação E_8 , vista na Figura 5.21, preocupa-se em não permitir que a esteira de alimentação avance sem a existência de peças sobre ela. No estado 1 não há peças sobre a esteira de alimentação, sendo proibido que esta última avance – evento I_{ea} . O evento I_{ea} somente é habilitado no estado 2, quando há certeza que existe pelo menos uma peça sobre a esteira de alimentação, após ocorrer pelo menos um dos eventos $F_{col_A_ea}$, $F_{col_BD_ea}$, $F_{col_C_ea}$, F_{Aduana_A} , F_{Aduana_BD} ou F_{Aduana_C} . O auto-laço em E_8 , permite que os eventos listados anteriormente possam ocorrer mais de uma vez, possibilitando a execução concorrente destas tarefas.

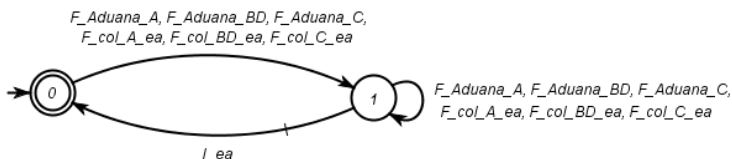


Figura 5.21 Especificação do avanço da esteira E_8

As especificações E_{9a} , E_{9b} e E_{9c} , modeladas como na Figura 5.22, tratam a exclusão mútua que deve haver entre a esteira de alimentação e o robô XR4, bem como entre a aduana e o robô SCARA, durante a execução do sistema. A partir destes geradores pode-se afirmar que uma vez iniciado o avanço da esteira de alimentação através do comando I_{ea} , não pode haver a execução de tarefas por parte dos robôs e da aduana, até que o evento F_{ea} seja sinalizado e vice-versa.

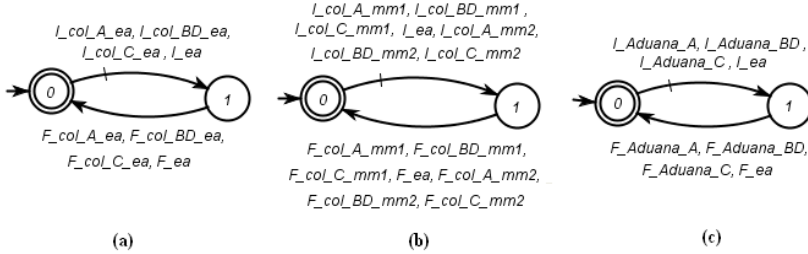


Figura 5.22 Especificações do mutex do robô XR4 e SCARA com a esteira de alimentação (a) E_{9a} , (b) E_{9b} e da aduana com a esteira de alimentação (c) E_{9c}

A montagem do conjunto final é feita pelo robô SCARA em duas posições diferentes da mesa de montagem. Para isto, são modeladas duas especificações E_{10a} e E_{10b} , as quais podem ser vistas na Figura 5.23 (a) e (b) respectivamente. A especificação E_{10a} responde pela montagem de peças na posição 1 e a E_{10b} na posição 2 da mesa de montagem. Cada gerador tem 5 estados que definem a sequência de montagem do conjunto final. Ressalta-se que a montagem sempre deve começar com uma base com um disco incluso; seguida de um anel e um cilindro em qualquer ordem, porém só uma unidade de cada. Finalmente o conjunto pronto é depositado na esteira de saída através do comando $I_{col_Pronta1_es}$ ou $I_{col_Pronta2_es}$, se o conjunto é oriundo da posição 1 ou da posição 2 da mesa de montagem, respectivamente.

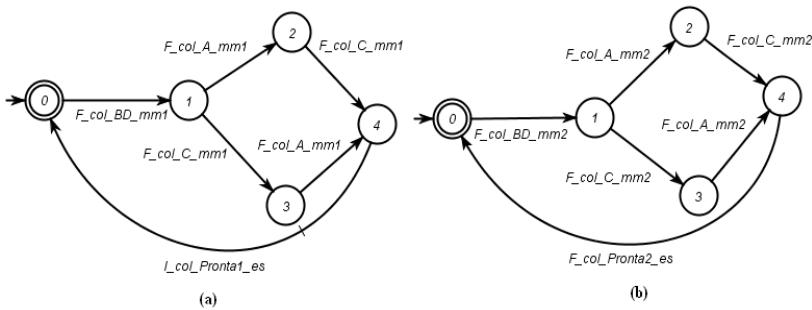


Figura 5.23 Especificações da montagem de conjuntos na posição 1 (a) E_{10a} e na posição 2 (b) E_{10b} da mesa de montagem

O processo de montagem de um conjunto final exige a prioridade de uma base com disco, sobre um anel ou um cilindro, conforme mencionado na seção 5.1, caso contrário, o robô SCARA não inicia a montagem e o sistema fica bloqueado. Além disso, é possível encontrar se-

quências de eventos que gerem também bloqueio ao deixar entrar no torno, um anel ou um cilindro. Por exemplo, depois de ter ido uma base com disco e um cilindro para a esteira de alimentação, a peça que faltaria para terminar a montagem seria um anel, mas a peça que está no torno é um cilindro. O cilindro poderia ser mesmo rejeitado pelo sistema de visão dando passo a um anel entrar no torno. No entanto, o evento $I_{col_A_tcnc}$ está desabilitado por não ter anel disponível no seu alimentador e o evento $I_{col_C_tcnc}$ está habilitado novamente, o que causa um bloqueio.

Para resolver estes problemas, foi modelado o gerador E_{11} da Figura 5.24, o qual reconhece uma linguagem onde sempre após uma base com disco, garante-se um anel e um cilindro. Não é considerado o sistema de visão, mas sim o torno para deixar entrar um anel ou um cilindro ao sistema. Na especificação E_{11} , o robô XR4 é desabilitado para pegar anéis – evento $I_{col_A_tcnc}$ – ou cilindros – evento $I_{col_C_tcnc}$ – sem antes haver aprovado uma base no sistema de visão e posteriormente ter colocado sobre ela um disco – evento $F_{col_D_bsv}$. Também, no caso de ser rejeitado um anel – evento $I_{col_A_nok}$ – ou um cilindro – evento $I_{col_C_nok}$ – habilita-se um novo ingresso destas duas peças, limitando sua quantidade a uma peça de cada tipo por cada base aprovada.

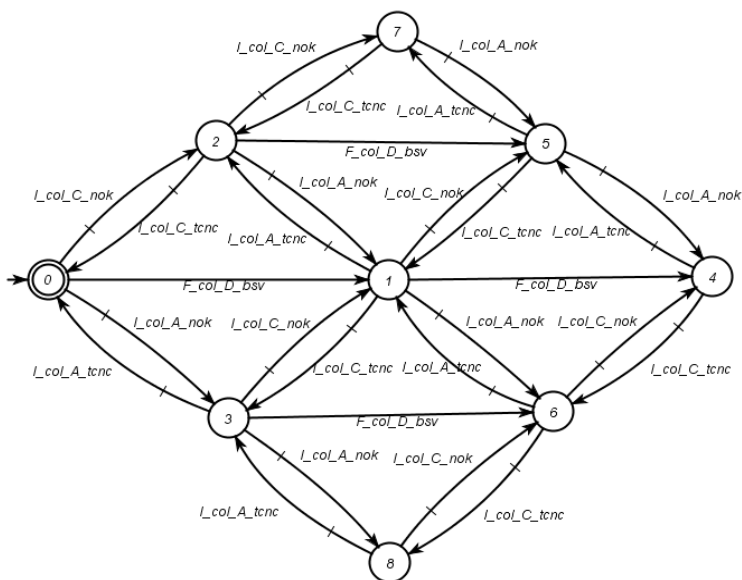


Figura 5.24 Especificação da prioridade das bases sobre os anéis e os cilindros E_{11}

Conforme mencionado anteriormente, quando uma base com disco é acompanhado por um anel e um cilindro, é relevante deixar entrar mais do que duas bases ao sistema. Mas para o caso onde temos, por exemplo, 3 bases habilitadas para entrar ao sistema, assim que a terceira base ocupa o sistema de visão, ocorre um bloqueio. Então, é necessário um modelo que limite as bases, como acontece com a especificação E_{12} , ilustrado na Figura 5.25. A diferencia da especificação E_{11} , a qual limita a entrada ao torno de um anel e um cilindro por cada base aceita, o gerador E_{12} limita em duas, a quantidade de bases que podem ser depositadas no centro de usinagem – evento $I_col_B_us$.

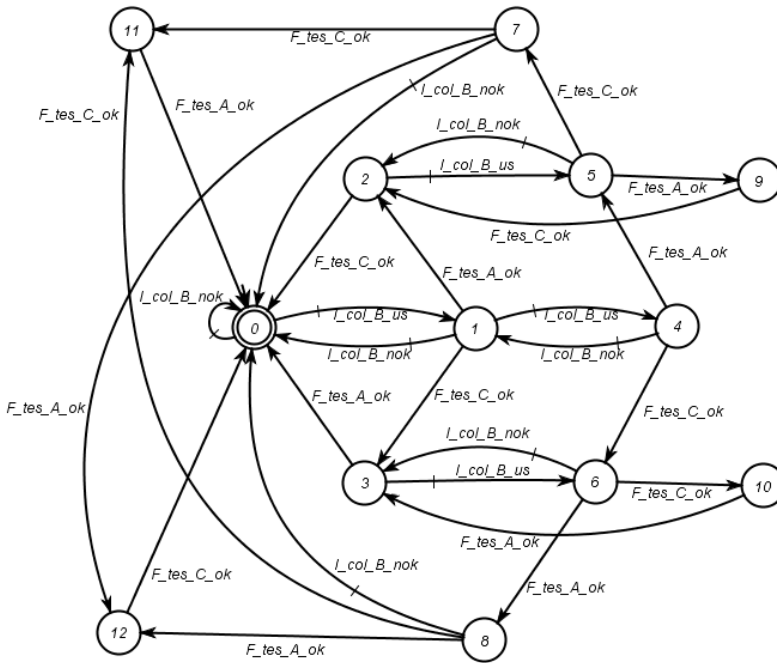


Figura 5.25 Especificação de limite de duas bases E_{12}

Uma vez que as duas bases tenham sido aprovadas, o evento $I_col_B_us$ é desabilitado até não ter ocorrido o último evento $F_tes_A_ok$ ou $F_tes_C_ok$ que completa a sequência de eventos para obter dois conjuntos prontos, independente de que um deles tenha sido colocado na esteira de saída. Esta restrição é imposta pelo fato de que pode acontecer uma sequência que leve a um bloqueio. Suponha que foram aprovadas duas bases que se encontram nas posições 1 e 2 respec-

tivamente da mesa de montagem. Em seguida um anel e um cilindro são aprovados. Uma opção pode ser que estas últimas façam parte da base que está na posição 1, para ter como resultado um conjunto final pronto para ser depositado na esteira de saída. Neste momento, seria mais flexível permitir a entrada de outra base no sistema. Porém, outra escolha pode ser tomada: o anel para a posição 1 e o cilindro para a posição 2, fazendo que o robô SCARA espere por outro anel e outro cilindro para completar a montagem. Assim, ao permitir o ingresso de outra base que fosse aprovada, causaria propriamente um bloqueio. O evento $I_{col_B_us}$ também pode ser habilitado novamente toda vez que uma base é reprovada no sistema de visão. Juntas, as especificações E_{11} e E_{12} colaboram na resolução do bloqueio por sequência de eventos que não obedecem a uma montagem correta de peças.

A especificação E_{13} , ilustrada na Figura 5.26, preocupa-se em não permitir que a esteira de saída avance sem a existência de um conjunto final sobre ela. No estado 1 não há conjuntos sobre a esteira de saída, sendo proibido que a esteira de saída avance – evento I_{es} . O evento I_{es} somente é habilitado no estado 2, quando é certo que um conjunto final está sobre a mesa, após ocorrer os eventos $F_{col_Pronta_1_es}$ ou $F_{col_Pronta_2_es}$.

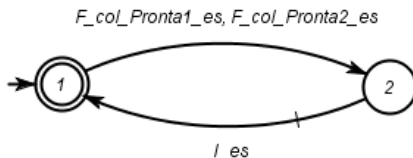


Figura 5.26 Especificação esteira de saída E_{13}

5.2.3 Etapa 2.3 Síntese de Supervisores

Conforme foi apresentado na seção 3.2, na abordagem modular local são exploradas as modularidades das especificações e da planta. As plantas locais são obtidas a partir da composição síncrona dos geradores dos subsistemas afetados pelas restrições locais, ou seja, aqueles que compartilham ao menos um evento a especificação.

Tomando como exemplo a construção da planta local G_{loc8} , referente à E_8 , tem-se que o alfabeto desta especificação é definido por $\Sigma_{E8} = \{F_{col_A_ea}, F_{col_BD_ea}, F_{col_C_ea}, F_{Aduana_A}, F_{Aduana_BD}, F_{Aduana_C}, I_{ea}\}$. Comparando este alfabeto com os alfabetos dos geradores dos subsistemas da planta, o qual pode ser obti-

do a partir das tabelas 5.1 e 5.2, nota-se que os geradores G_8 , G_9 e G_{10} apresentam eventos comuns à especificação E_8 . Desta forma, a planta local é a composição síncrona dos geradores G_8 , G_9 e G_{10} . As plantas locais para a CFM são vistas na tabela 5.2.

Tabela 5.2 Plantas locais para o sistema RHINO

G_{locx}	$\parallel G_x$
G_{loc1a}	$G_1 \parallel G_8$
G_{loc1b}	$G_2 \parallel G_8$
G_{loc1c}	$G_3 \parallel G_8$
G_{loc1d}	$G_4 \parallel G_8$
G_{loc2a}	G_8
G_{loc2b}	G_8
G_{loc2c}	G_8
G_{loc3a}	$G_5 \parallel G_8$
G_{loc3b}	$G_7 \parallel G_8$
G_{loc3c}	$G_5 \parallel G_8$
G_{loc4a}	$G_5 \parallel G_8$
G_{loc4b}	$G_7 \parallel G_8$
G_{loc4c}	$G_5 \parallel G_8$
G_{loc5a}	$G_6 \parallel G_8$
G_{loc5b}	$G_6 \parallel G_8$
G_{loc5c}	$G_6 \parallel G_8$
G_{loc6a}	$G_6 \parallel G_8$
G_{loc6b}	$G_6 \parallel G_8$
G_{loc6c}	$G_6 \parallel G_8$
G_{loc7a}	$G_8 \parallel G_9 \parallel G_{10}$
G_{loc7b}	$G_9 \parallel G_{10} \parallel G_{11}$
G_{loc8}	$G_8 \parallel G_9 \parallel G_{10}$
G_{loc9a}	$G_8 \parallel G_9$
G_{loc9b}	$G_9 \parallel G_{11}$
G_{loc9c}	$G_9 \parallel G_{10}$
G_{loc10a}	G_{11}
G_{loc10b}	G_{11}
G_{loc11}	G_8
G_{loc12}	$G_6 \parallel G_8$

Conforme visto, o cálculo da linguagem alvo K_{locx} para cada planta local G_{locx} é definida por $K_{locx} = G_{locx} \parallel E_x$. Então aplicado ao sistema RHINO tem-se que $x \in X = \{1a, 1b, 1c, 1d, 2a, 2b, 2c, 3a, 3b, 3c, 4a, 4b, 4c, 5a, 5b, 5c, 6a, 6b, 6d, 7a, 7b, 8, 9a, 9b, 9c, 10a, 10b, 11, 12\}$. Em seguida é calculada a máxima linguagem controlável contida em cada linguagem alvo, $S_{locx} = (K_{locx}, G_{locx})$. Para o sistema RHINO foram obtidos 30 supervisores S_{locx} , $x \in X$. Devido à explosão de estados decorrente da alta complexidade atribuída à ampla gama de tarefas de coordenação entre os diversos subsistemas, associadas a cada especificação, não foi possível realizar o teste de modularidade local. As ferramentas TCT e Supremica não suportam esta operação. Porém, conseguiu-se realizar com a ferramenta TCT o produto síncrono de 23 especificações, $S = S_{loc1a} \parallel S_{loc1b} \parallel S_{loc1c} \parallel S_{loc1d} \parallel S_{loc2a} \parallel S_{loc2b} \parallel S_{loc2c} \parallel S_{loc3a} \parallel S_{loc3b} \parallel S_{loc3c} \parallel S_{loc4a} \parallel S_{loc4b} \parallel S_{loc4c} \parallel S_{loc5a} \parallel S_{loc5b} \parallel S_{loc5c} \parallel S_{loc6a} \parallel S_{loc6c} \parallel S_{loc6d} \parallel S_{loc7a} \parallel S_{loc11} \parallel S_{loc12}$. O gerador resultante apresenta 634784 estados e 4595152 transições e é *trim*, o que significa que a ação conjunta destes supervisores é não bloqueante pelo menos até a entrada de peças na esteira de alimentação. Entretanto, isto não quer dizer que o bloqueio está resolvido, visto que o problema de conflito é um problema global que ocorre pela interação de todos os supervisores com toda a planta.

Posteriormente, foram reduzidos os supervisores para diminuir o uso de memória computacional e tornar compreensível a interpretação dos supervisores na etapa de implementação no CLP. O número de estados de todos os geradores envolvidos na síntese de supervisores modulares locais é visualizado na tabela 5.3. Observa-se que os geradores dos supervisores reduzidos obtidos em alguns casos são equivalentes às especificações genéricas.

Tabela 5.3 Número de estados dos geradores da síntese dos supervisores

x	E_x	G_{locx}	E_{locx}	S_{locx}	RS_{locx}
1a, 1b, 1c, 1d	2	14	28	28	2
2a, 2b, 2c	2	14	26 22 27	24 18 26	2
3a, 3b, 3c	2	42 28 42	84 56 84	81 54 81	2
4a, 4b, 4c	2	42 28 42	84 56 84	70 42 70	2

5a, 5b, 5c	2	56	112	108	2
6a, 6b, 6c	3	56	168	140	3
			160	127	4
			168	140	3
6d	2	14	112	94	2
7a	16	112	1792	1504	16
7b	16	72	1152	504	16
8	2	112	224	224	2
9a, 9b, 9c	2	28	25	25	2
		18	12	12	
		8	5	5	
10a, 10b	5	9	41	31	5
			41	31	
11	9	14	114	109	9
12	13	56	692	584	24
13	2	18	36	32	2

5.3 FASE III: EMULAÇÃO

Como mencionado na seção 5.1, a emulação do funcionamento da planta sob controle dos supervisores reduzidos obtidos na etapa 2.3, foi realizada pela ferramenta Supremica. Através dessa fase, identificou-se a necessidade de alteração nos modelos de algumas especificações e da construção de novas especificações que resolvessem problemas de bloqueio relacionados com a coordenação das operações do sistema RHINO. O ciclo de emulação foi repetido inúmeras vezes, tendo que voltar à fase anterior para realizar as respectivas correções. Ele foi encerrado quando o sistema RHINO estava seguindo a lógica de controle proposta.

5.4 FASE IV: IMPLEMENTAÇÃO DO CONTROLE SUPERVISÓRIO EM CLP

A ferramenta Ides2ST gerou automaticamente o código do controle supervisório em linguagem de texto estruturado, para ser implementado no CLP, recebendo como entrada os arquivos da planta e dos supervisores na forma de geradores construídos na ferramenta IDES. O código gerado, como apresentado no capítulo 3, é dividido em três partes: o código que implementa os supervisores, as desabilitações e o sis-

tema produto. Cada trecho de código é inserido em módulos de procedimento do CLP Altus através da sua ferramenta de programação MasterTool. Posteriormente, são implementadas em linguagem *ladder* as sequências operacionais que implementam os subsistemas: alimentadores, estações de trabalho, aduana e esteiras. Para os robôs XR4 e SCARA, é implementado um código em linguagem de texto estruturado que serve de interface com os seu controladores. Finalmente, é realizado um teste inicial de funcionamento do sistema, encontrando-se erros de programação nas sequências operacionais dos alimentadores e do sistema de visão associado ao RANDOM que simula a operação de teste. Estes problemas foram resolvidos.

5.5 FASE V: IMPLEMENTAÇÃO DE FUNCIONALIDADES BÁSICAS DO SISTEMA SCADA

É criada uma nova aplicação, inserindo as *tags* relacionadas com as variáveis do código gerado na fase anterior e estabelecendo a comunicação Modbus RTU com o CLP Altus. Não é necessário instalar e configurar o software SacadBR, já que esse processo foi realizado para a CFM do capítulo 4. Dessa forma se deu início à implementação de funcionalidades básicas do sistema SCADA.

O sinótico é implementado através de objetos gráficos associados aos estados dos geradores do sistema produto. De igual forma como o exemplo do capítulo 4, é encontrado que esses estados não conseguem mostrar toda a informação necessária e suficiente do processo. Essa informação está relacionada à permanência de peças nas estações de trabalho uma vez terminadas suas operações correspondentes e ao percurso de peças pela esteira de alimentação. Portanto, são descritos os estados dos 30 supervisores, determinando no final que aquela informação é fornecida por alguns estados dos supervisores 4A, 4B, 4C, 6A, 6B, 6C, 6D, 7A e 7B. Na tabela 5.4, são descritos esses estados para serem incluídos posteriormente na implementação do sinótico. Cabe mencionar que os estados que têm mais de um significado associado a situações diferentes, não são incluídos no sinótico.

Na Figura 5.27 é apresentado o sinótico obtido com as informações anteriores, no qual podem ser observados quadros com os nomes dos geradores do sistema produto e dos supervisores utilizados, como também alguns objetos gráficos que aparecem quando um estado é ativado.

Tabela 5.4 Descrição dos estados relevantes dos supervisores para o sinótico do sistema RHINO

Supervisor	Estado	Significado
4A	1	Anel torneado
4B	1	Base usinada
4C	1	Cilindro torneado
6A	½	Anel reprovado/aprovado
6B	2/3	Base aprovada com disco no sistema de visão/Base reprovada.
6C	½	Cilindro reprovado/aprovado.
6D	1	Base aprovada esperando disco no sistema de visão.
7A	1/2/3	Anel/Base/Cilindro em posição 1 da esteira de alimentação.
7A	4/5/6	Anel/Base/Cilindro indo da posição 1 à posição 2 da esteira de alimentação.
7B	1/2/3	Anel/Base/Cilindro em posição 2 da esteira de alimentação.
7B	4/5/6	Anel/Base/Cilindro indo da posição 1 à posição 2 da esteira de alimentação.
10A/10B	2	Base com disco e anel na posição 1/posição 2 da mesa de montagem.
10A/10B	3	Base com disco e cilindro na posição 1/posição 2 da mesa de montagem.
10A/10B	4	Conjunto pronto na posição 1/posição 2 da mesa de montagem.
13	1	Conjunto pronto na esteira de saída.

Suponha que o sistema está operando em condições normais e os quadros em cor verde significam o gerador ativo. Observa-se na Figura 5.27, que o centro de usinagem, associado ao gerador G_7 , está operando uma base (estado 1), representado por um círculo verde pequeno nessa estação. Por outro lado, o sistema de visão (gerador G_6) reprovou um anel (estado 1 do supervisor S6A) representado por um componente gráfico em cor vermelha. Posteriormente essa peça é transportada pelo robô XR4 (gerador G_8) até o *buffer* de descarte (estado 9), ao passo que o torno (gerador G_5) finaliza sua operação e o cilindro, em cor amarela, espera para ser transportado para o sistema de visão (estado 1 do supervisor S4C).

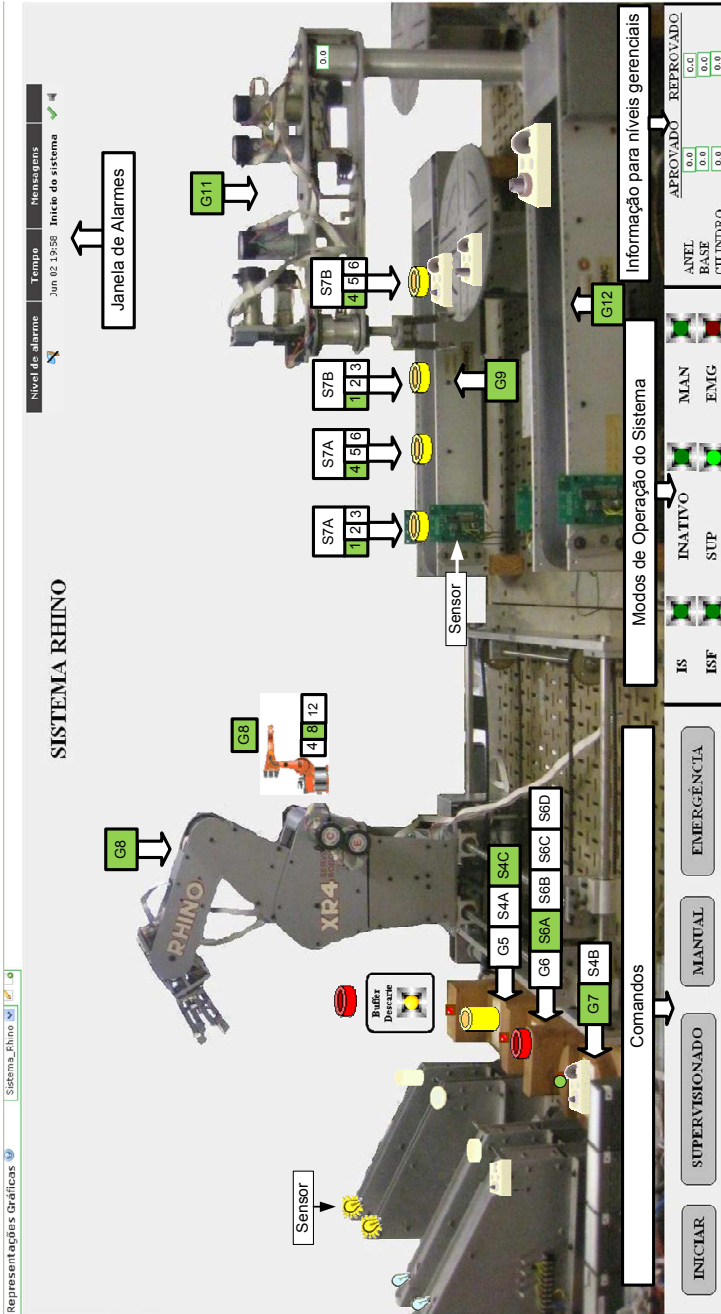


Figura 5.27 Sinótico para o sistema RHINO

Quando um anel é aprovado, aparece um robô pequeno, animado e em cor laranja, indicando que o robô XR4 o está transportando até a esteira de alimentação (estado 8 do gerador G_8). Terminada esta operação, é visualizado o percurso do anel sobre a esteira de alimentação (gerador G_9), relacionado aos estados 1 e 4 dos supervisores 7A e 7B.

O sinótico mostra ainda a montagem de peças nas duas posições da mesa de montagem realizada pelo robô SCARA (gerador G_{11}) associada aos supervisores 10A e 10B. O funcionamento das esteiras é representado por um componente animado associado aos geradores G_9 e G_{12} . Por sua vez, o conjunto final na esteira de saída é também representado por um componente gráfico associado ao supervisor S13. O sinótico é finalizado inserindo janela de alarmes, comandos para inicialização dos modos de operação e ícones que permitem visualizar informação para níveis gerenciais.

Os comandos também foram implementados em linguagem de texto estruturado de forma semelhante ao explicado na seção 4.2.2. Quando o sistema encontra-se no estado Manual, os eventos controláveis associados a cada subsistema podem ser ativados através dos eventos *forca Equipamento* inseridos no código do sistema produto, expressos por comandos na tela implementada para tal fim, que é visualizada na Figura 5.28.

O histórico de eventos para o sistema RHINO é implementado da mesma forma como no exemplo da CFM do capítulo 4. A Figura 5.29 ilustra um histórico de eventos obtido com a aplicação do método apresentado na seção 4.2.3.

A geração de alarmes críticos segue a mesma sistemática da apresentada na seção 4.2.4. Na Figura 5.30 é visualizado um trecho de código do sistema produto onde é implementado o evento controlável I_{tor_C} do gerador G_5 que representa o torno. Suponha o sistema no estado Manual e as variáveis *unsafe* e *De_I_tor_C* em zero. Quando é ativado o evento I_{tor_C} , o gerador G_5 passa do estado 0 ao estado 2. Nesse momento seu evento de desabilitação *De_I_tor_C* é ativado, isto é, o evento I_{tor_C} não pode acontecer novamente. Se isso ocorrer a partir da ativação do evento *forca_tor_c*, a variável *unsafe* toma o valor da desabilitação, logo é gerado um alarme com o nome *sistema não seguro*, o que indica que o operador violou alguma especificação de controle, significando comportamento não desejado do sistema. Esse alarme deve ser reconhecido pelo operador a través da janela de alarmes, o que não implica que o sistema volte a um estado seguro. Nesta condição o sistema deve ser reinicializado. As 4 funcionalidades implementa-

das nesta fase foram verificadas em sua função até obter um desempenho satisfatório.


TORNO	RHINO	SCARA
ANEL + CILINDRO	ANEL	ANEL
▶ Tornear Anel	▶ Colocar Anel no Torno	▶ Colocar Anel na posição 1 da Mesa de Montagem
▶ Tornear Cilindro	▶ Colocar Anel no Sistema de Visão	▶ Colocar Anel na posição 2 da Mesa de Montagem
CENTRO DE USINAGEM	▶ Colocar Anel no Buffer de Descarte	BASE + DISCO
BASE + DISCO	▶ Colocar Anel na Esteira de Alimentação	▶ Colocar Base + Disco na posição 1 da Mesa de Montagem
▶ Fresar Base	BASE + DISCO	▶ Colocar Base + Disco na posição 2 da Mesa de Montagem
SISTEMA DE VISÃO	▶ Colocar Base no Centro de Usinagem	CILINDRO
ANEL + BASE + CILINDRO	▶ Colocar Base no Buffer de Descarte	▶ Colocar Cilindro na posição 1 da Mesa de Montagem
▶ Testar Anel	▶ Colocar Base + Disco na Esteira de Alimentação	▶ Colocar Cilindro na posição 2 da Mesa de Montagem
▶ Testar Base	▶ Colocar Base no Sistema de Visão	ANEL + BASE + CILINDRO
▶ Testar Cilindro	▶ Colocar Disco no Sistema de Visão	▶ Colocar Conjunto da posição 1 na esteira de saída
ESTEIRA DE ALIMENTAÇÃO	CILINDRO	▶ Colocar Conjunto da posição 2 na esteira de saída
▶ Avanço Esteira de Alimentação	▶ Colocar Cilindro no Torno	ESTEIRA DE SAÍDA
ADUANA	▶ Colocar Cilindro no Sistema de Visão	▶ Avanço Esteira de Saída
▶ Início Aduana Anel	▶ Colocar Cilindro no Buffer de Descarte	
▶ Início Aduana Base + Disco	▶ Colocar Cilindro na Esteira de Alimentação	
▶ Início Aduana Cilindro		
SISTEMA NÃO SEGURO 		

Figura 5.28 Tela de comandos estado Manual

Histórico (as of 2011/07/10 00:35:23)	
Valor	Tempo
F_col_B_nok: Término do transporte de peça base d	00:35:19
I_col_B_nok: Início do transporte de peça base do s	00:35:12
F_tes_B_nok: Término de teste indicando que a peç	00:35:12
I_tes_B: Início de teste de peça base	00:35:07
F_col_B_sv: Término do transporte de peça base do	00:35:07
I_col_B_sv: Início do transporte de peça base do ce	00:34:59
F_us_B: Término de fresagem da peça base	00:34:59
I_us_B: Início de fresagem de peça base	00:34:54
F_col_B_us: Término do transporte de peça base d	00:34:54
I_col_B_us Início do transporte de peça base do se	00:34:41
F_col_B_nok: Término do transporte de peça base d	00:33:13
I_col_B_nok: Início do transporte de peça base do s	00:32:37
F_tes_B_nok: Término de teste indicando que a peç	00:32:37

Figura 5.29 Histórico de eventos gerado para o sistema RHINO

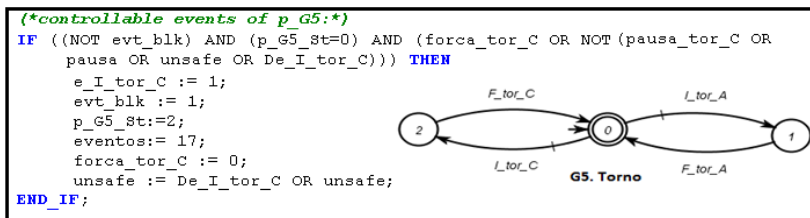


Figura 5.30 Trecho de código do evento controlável I_tor_C

5.6 FASE VI: AVALIAÇÃO DO FUNCIONAMENTO DO SISTEMA REAL

Nesta fase são conectados os dispositivos de campo ao CLP Altus. Em seguida são realizados testes de avaliação do funcionamento do sistema RHINO, auxiliados pelas funcionalidades implementadas na fase V, tais como: o sinótico, os comandos, o histórico de eventos e os alarmes críticos, encontrando-se novamente erros de programação, desta vez na sequência operacional dos robôs XR4 e SCARA e na implementação do modo Manual, os quais puderam ser corrigidos voltando à fase anterior. Cabe ressaltar que o uso do histórico de eventos foi decisivo para facilitar a resolução dos problemas mencionados.

5.7 VII: IMPLEMENTAÇÃO DE FUNCIONALIDADES DO SISTEMA SCADA

Para o sistema RHINO, são implementados 4 alarmes gerais; 3 referentes à disponibilidade de peças nos alimentadores de anéis, bases e cilindros e outro referente à presença de peça na posição 1 da esteira de alimentação. Os alarmes dos alimentadores são gerados a partir do sinal dos sensores de toque associados a eles. Já o alarme da esteira, é gerado quando o sinal do sensor da primeira posição não é ativado quando os estados 1, 2 ou 3 do gerador S7A, os quais identificam uma peça nessa posição como pode ser visto na tabela 5.4, dizem o contrário. A Figura 5.31 apresenta o código que implementa este alarme.


```
(*Alarme Geral*)  
  
IF NOT s_ea AND (s_REDUIZIDO7A_St =1 OR s_REDUIZIDO7A_St =2 OR s_REDUIZIDO7A_St =3) THEN  
  alarme_ea :=1;  
ELSE  
  alarme_ea:= 0;  
END_IF;
```

Figura 5.31 Código que implementa o alarme da esteira de alimentação

Conforme mencionado, o sistema RHINO fabrica um produto constituído de 4 peças. Então é implementada uma receita para determinar a quantidade de produtos que deve ser fabricada. Isto é feito com um código em linguagem de texto estruturado, apresentado na Figura 5.32, que faz uma contagem de conjuntos prontos toda vez que um deles é depositado na esteira de saída, através dos eventos *e_F_col_Pronta1_es* e *e_F_col_Pronta2_es*. Suponha que o operador do sistema digite 4 no campo *N de conjuntos a fabricar* na tela da receita, Figura 5.33, o qual está associado à variável *receita_scada*. Esse valor é comparado com o valor mostrado no campo *Conjuntos fabricados*, que no código está associado à variável *num_conjunto_pronto*. Se os valores são iguais, a variável *pausa*, discutida anteriormente, é ativada, não permitindo a fabricação de mais conjuntos até ativar o campo *Nova Receita* que coloca em zero a variável *pausa*, possibilitando a geração de uma nova receita para o processo.

```
(*Receita*)  
  
IF (e_F_col_Pronta1_es OR e_F_col_Pronta2_es) THEN  
  num_conjunto_pronto:= num_conjunto_pronto + 1;  
END_IF;  
  
IF (num_conjunto_pronto<>0) AND (receita_scada = num_conjunto_pronto) THEN  
  pausa := 1;  
  num_conjunto_pronto :=0;  
  receita_scada:=0;  
END_IF;
```

Figura 5.32 Código que implementa uma receita no sistema RHINO

The image shows a graphical user interface for a recipe management system. At the top, there is a header box containing the text 'RECEITA' and 'SISTEMA RHINO'. Below this header, there are three distinct input areas: a box for 'Nº Conjuntos a fabricar', a box for 'Conjuntos fabricados', and a button labeled 'Nova Receita'.

Figura 5.33 Tela da receita

Uma informação que pode ser relevante para este processo é saber quantas peças foram aprovadas e reprovadas durante um ciclo normal de operação. Então, são inseridas seis novas variáveis no código do sistema produto, na parte de eventos não controláveis, chamadas *contador_anel_a*, *contador_base_a*, *contador_cilindro_a*, *contador_anel_r*, *contador_base_r* e *contador_cilindro_r*, as quais realizam a contagem de anéis, bases e cilindros aprovados e reprovados respectivamente, quando os eventos *F_tes_a_ok*, *F_tes_b_ok*, *F_tes_c_ok*, *F_tes_a_nok*, *F_tes_b_nok*, *F_tes_c_nok* são ativados. Na Figura 5.34 é visualizado um trecho de código que implementa a contagem de anéis aprovados e reprovados. A Figura 5.35 mostra as tendências para as variáveis mencionadas acima.

```

IF ((NOT evt_blk) AND (p_G6_St=1) AND Ae_F_tes_A_ok) THEN
  e_F_tes_A_ok := 1;
  Ae_F_tes_A_ok := 0;
  evt_blk := 1;
  p_G6_St:=0;
  eventos:= 39;
  contador_anel_a:= contador_anel_a +1;
END_IF;

IF ((NOT evt_blk) AND (p_G6_St=1) AND Ae_F_tes_A_nok) THEN
  e_F_tes_A_nok := 1;
  Ae_F_tes_A_nok := 0;
  evt_blk := 1;
  p_G6_St:=0;
  eventos:= 40;
  contador_anel_r:= contador_anel_r +1;
END_IF;

```

Figura 5.34 Trecho de código contagem de anéis aprovados e reprovados

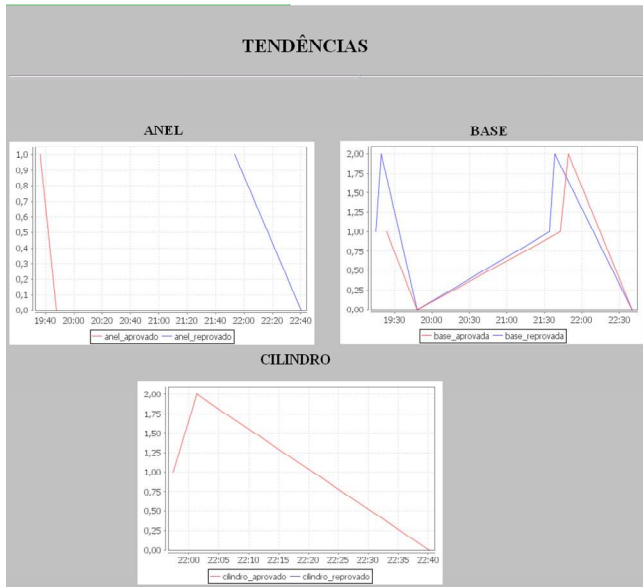


Figura 5.35 Tendências de aprovação e reprovação de peças

O exemplo de relatório apresentado aqui é o mesmo gerado para a CFM. A Figura 5.36 mostra parte do relatório gerado para a variável *descricao_estados* que representa o histórico de eventos para o sistema RHINO.

User access		Username		AccessType	
		admin		admin	
Data point		Eventos			
Meta_Auxiliar-descricao_estados		Todos os eventos			
Histórico		Views			
Valor	Tempo	Nome			
I_col_C_tenc: Início do transporte de peça cilindro d	22:17:13	Data point não utilizado em nenhuma de suas representações gráficas			
I_ea: Início do avanço da esteira de alimentação	22:03:03				
F_col_C_ea: Término do transporte de peça cilindro	22:02:38				
F_col_C_sv: Término do transporte de peça cilindro	22:01:12				
I_Aduana_C: Início da aduana de peça cilindro	22:00:24				
F_tor_C: Término de torneamento de peça cilindro	21:58:42				
F_tes_C_ok: Término de teste indicando que a peça	21:57:22				
F_col_A_nok: Término do transporte de peça anel d	21:56:13				
Chega_C: Chega peça cilindro no seu alimentador	21:54:07				
F_col_A_sv: Término do transporte de peça anel do	21:53:28				
I_Aduana_BD: Início da aduana do conjunto base +	21:52:38				
I_col_BD_ea: Início do transporte do conjunto base	21:50:35				
I_tes_B: Início de teste de peça base	21:49:17				
I_col_B_sv: Início do transporte de peça base do ce	21:47:53				
		Eventos			
		Id			
		Nível de alarme			
		Sem eventos para listar			

Figura 5.36 Relatório do histórico de eventos do sistema RHINO

5.8 FASE VIII: VALIDAÇÃO

O ciclo de desenvolvimento desta metodologia é encerrado realizando testes finais de operação do sistema RHINO com todas as funcionalidades do SCADA implementadas, as quais fornecem uma visão global do sistema em funcionamento. O sistema é submetido a inúmeros ciclos de fabricação de conjuntos de peças sem encontrar nenhum tipo de bloqueio, determinando assim que a coordenação entre os subsistemas está de acordo com as leis de controle especificadas. Ao final do processo, observa-se um sistema SCADA integrado ao CLP, funcionando de forma satisfatória.

5.9 ANÁLISE DE RESULTADOS

A aplicação da metodologia proposta no capítulo 4 para o sistema RHINO permitiu um desenvolvimento sistemático, flexível e eficiente do projeto de controle em CLP integrado ao sistema SCADA. Isso se dá pelo fato de ser implementada seguindo os procedimentos que caracterizam cada uma de suas fases, ao passo de tornar possível a realização de reconfiguração e alteração na lógica de controle de forma rápida e pontual, como também a eficácia advinda da aplicação de testes para detecção de erros.

A estruturação da lógica de controle na forma de geradores permite que informações do sistema real sejam coletadas a partir dos estados dos modelos das plantas e supervisores implementados no CLP, mesmo na ausência de sensores que forneçam diretamente essas informações. Assim a aplicação das especificações de controle sobre o sistema pode ser visualizada em tempo real. O monitoramento completo, controle e distribuição dos dados de toda a planta contribuem para a tomada de decisões produtivas.

Além do mais, a integração das funcionalidades do sistema SCADA com o código de controle supervísório permitiu a identificação e correção de erros de programação e digitação. Estas funcionalidades também permitiram tanto a visualização do modo de operação do sistema (inicialização de software, inicialização do sistema físico, inativo, supervisionado, manual e emergência) por meio do sinótico, quanto à condução do sistema ao estado inicial, supervisionado e manual, através de comandos.

De forma similar foi possível a geração seletiva dos eventos controláveis no estado manual, associados aos subsistemas do sistema RHINO. Nesse estado, são gerados alarmes ligados ao estado ativo da variável *unsafe*, indicando que o sistema tem violado alguma especificação de controle. Também foram implementadas 3 alarmes combinando os estados 1, 2 e 3 do supervisor 7A com o sensor da esteira de alimentação para sinalizar a ausência de peça sobre a esteira num processo normal de operação.

Por outro lado, foi possível a geração de receitas baseadas na informação inserida na representação gráfica da receita que se encontra na aplicação SCADA, e dos eventos do fim do depósito de conjuntos finais sobre a esteira de saída. Também foram geradas tendências, históricos e relatórios dos eventos controláveis e não controláveis para o tratamento de falhas.

Infelizmente, o teste de modularidade local não pôde ser realizado completamente devido à grande quantidade de supervisores envolvidos no controle do sistema RHINO, o que leva a uma explosão do espaço de estados e representa um problema recorrente no controle modular local e na TCS como um todo. Entretanto, conseguiu-se realizar o produto síncrono de 23 supervisores locais. O gerador resultante foi *trim* o que significa que a ação conjunta destes supervisores é não bloqueante pelo menos até a entrada de peças na esteira de alimentação. Isso não garante ausência de bloqueio, após todas as demais especificações serem atendidas.

6. CONCLUSÕES

Os sistemas SCADA atendem às necessidades industriais, visto que o monitoramento completo, controle e distribuição dos dados de toda planta auxilia nas tomadas de decisões mais rápidas, eficientes e produtivas, que permitem à empresa significativa vantagem competitiva. O sucesso depende muito da habilidade para acessar, entender e interpretar o grande volume de informações geradas pela operação do processo. E o monitoramento completo, controle e distribuição dos dados de toda planta auxilia nas tomadas de decisão mais rápidas, eficientes.

Existem na literatura acadêmica, poucas metodologias que tratam o desenvolvimento de Sistemas SCADA para sistemas de manufatura; algumas que se preocupam com a validação da implementação da lógica de controle em vários CLP e nenhuma que integre as funcionalidades do sistema SCADA ao controle supervísório programado em CLP. Este trabalho apresentou uma metodologia que abrange essa integração, aplicada ao sistema RHINO apresentado no capítulo 5, como também ao trabalho de Silva e Queiroz (2009) que já contava com uma lógica de controle definida pela ACS (QUEIROZ e CURY, 2002).

Esta metodologia utiliza a TCS em sua abordagem modular local, a programação da ACS em linguagem de texto estruturado e as funcionalidades oferecidas pelos Sistemas SCADA, para controlar e supervisionar um sistema de manufatura. Assim, foi proposto um método de implementação de funcionalidades de SCADA que fazem proveito da estrutura de implementação de controle supervísório modular local, as quais são relacionadas a seguir:

- Acompanhamento do processo de produção em tempo real e apresentação das informações por meio de interfaces gráficas baseadas nos estados dos geradores e no código que implementa o sistema produto e os supervisores;
- Execução de comandos para ativar os diversos modos de operação do sistema e ativação seletiva de eventos controláveis no modo manual, por parte do operador;
- Geração do histórico da sequência de eventos controláveis e não controláveis ocorridos durante o ciclo de operação normal, o qual permite detectar erros de programação e mau funcionamento de equipamentos e determinar causas de bloqueio;

- Geração e visualização de alarmes baseados na informação obtida de sensores e estados do sistema produto e supervisores indicando falhas do sistema;
- Execução de receitas e geração de informação relevante para os níveis gerenciais, baseadas igualmente em informação de estados e/ou eventos dos geradores;
- Geração de relatórios e tendências de qualquer variável associada ao controle supervisão que sirva de suporte na tomada de decisões sobre o processo produtivo e no tratamento de falhas.

A metodologia proposta é validada com a aplicação ao sistema RHINO seguindo todas suas fases. Uma limitação da aplicação da abordagem modular local encontrada particularmente neste caso, foi não poder realizar o teste de modularidade local devido à explosão do espaço de estados, resultante da composição síncrona de uma grande quantidade de supervisores locais, aumentando desta forma a complexidade computacional cujas ferramentas como TCT e Supremica não conseguiram resolver. Ainda assim, a abordagem modular local traz benefícios evidentes quando comparada aos procedimentos monolíticos. Além do mais, essa abordagem é decisiva quando se deseja introduzir ou alterar uma especificação, já que não é necessário realizar novamente a síntese de todo o conjunto de supervisores, mas apenas da parte específica que foi alterada ou introduzida. Além do mais, algumas alternativas que de certa forma fogem do escopo deste trabalho, poderiam ser adotadas para lidar especificamente com a complexidade do teste de modularidade, como por exemplo, o uso de abstrações (PENA et al., 2009).

Desta forma foi emulado o comportamento do sistema com os supervisores obtidos com a ferramenta Supremica, tendo como resultado um comportamento satisfatório. As funcionalidades do sistema SCADA foram implementadas com sucesso, possibilitando uma avaliação do funcionamento geral do sistema. Após aplicação da última fase da metodologia, concluiu-se que o sistema RHINO opera satisfatoriamente.

A ferramenta Ides2ST, baseada na estrutura genérica de controle supervisão, diminuiu o tempo de implementação do controlador, permitindo a geração automática de código em linguagem de texto estruturado (ST) de forma clara e sistemática. Estas características tornaram possível a integração do código do CLP com as funcionalidades do sistema SCADA. Por outro lado, o software ScadaBR usado na integração apresentou bastante problemas no início deste trabalho, associadas à comunicação serial Modbus com o CLP, além de não possuir muitas

opções para elaborar sinóticos e *scripts*. Isso atrasou a implementação preliminar que deu origem à metodologia proposta, devido a ser um software ainda em desenvolvimento. No entanto, com o decorrer do trabalho, mais especificamente no começo de 2011, os desenvolvedores do software foram adicionando mais protocolos, novas funcionalidades para *scripting*, novo construtor de telas e componentes gráficos, o que permite ser mais robusto na implementação de sistemas SCADA em processos industriais.

A aplicação desta metodologia confere sistematização, flexibilidade e eficiência ao projeto de controle e supervisão de sistemas de manufatura. Todo o processo de sistematização contribui para a diminuição do tempo necessário ao desenvolvimento de cada projeto lógico. Soma-se a esse fato, uma maior estruturação ao projeto de controladores, deixando a lógica de controle e sua programação mais clara para o projetista e os operadores do sistema, ao passo que permite a validação de soluções de controle.

Na continuidade deste trabalho pode-se citar a aplicação desta metodologia para gerenciar sistemas mais complexos distribuídos com CLP. Outra perspectiva para trabalhos futuros é a integração da estrutura proposta com os níveis superiores da pirâmide da automação de forma que as ferramentas de planejamento e programação da produção possam definir a escolha de eventos através do sistema SCADA, otimizando o uso de recursos do sistema na execução de múltiplas tarefas. Se o critério de otimização for o menor tempo de produção tem-se um problema de escalonamento da produção que poder ser resolvido através de heurísticas ou mesmo pelo uso da Teoria de Controle Supervisório. Pode-se ainda fazer a escolha de eventos de forma que os caminhos habilitados não levem a bloqueio, resolvendo assim o problema de conflito em supervisores modulares.

REFERÊNCIAS

- AKESSON, K. **Methods and Tools in Supervisory Control Theory. PhD Thesis, Chalmers University of Technology, Sweden, 2002.**
- ALMEIDA, F. R. M. **Sistemas SCADA e Aplicação.** Monografia (Graduação Engenharia Elétrica) – Universidade Federal do Ceará, Centro de Tecnologia, Fortaleza, 2009.
- BAILEY, D e WRIGHT, E. **Practical SCADA for Industry.** Perth, Australia, 2003.
- BALIEIRO, D. **Aplicação da Teoria de Controle Supervisório no projeto de controladores para sistemas de Rota variável centrado em robô PPGEPS.** Dissertação (Mestrado em Engenharia de Produção e Sistemas) – Pontifícia Universidade Católica do Paraná, Curitiba, 2007.
- BOARETTO, N. **Sistemas Supervisórios.** Instituto Federal Santa Catarina, Joinville, 2008.
- BOUZON, G; QUEIROZ, M.H. e CURY, J.E.R. Controle Supervisório de Sistemas a Eventos Discretos com Sensores Distinguidores.
- BOUZON, G.; de QUEIROZ, M. H. e Cury, J. E. **Supervisory Control of DES with Distinguishing Sensors.** Proc. 9th Workshop on Discrete Event Systems, Gothenburg, Sweden, pp. 390–391, 2008.
- BOYER, Stuart A. **SCADA: supervisory control and data acquisition.** ISA – Instrument Society of America, USA, 2004.
- BUSSETI, M. A.; SANTOS, E. A. P. **A project Methodology Applied to Automated and Integrated Manufacturing Systems.** Third International Conference on Production Research Americas“ Region, Curitiba, Brazil, 2006.
- CABÚS, J.R.; NAVARRETE, D.G. e PORRAS, R.P. **Sistemas SCADA.** Miniproyecto Automatización Industrial. Especialidad (Electrónica Industrial). Escola Politècnica Superior d’ Enginyeria de Vilanova i La Geltrú. Universitat Politècnica de Catalunya. 2004
- CASSANDRAS, G.C.; LAFORTUNE, S. **Introduction to Discrete Event Systems.** KLUWER Academic Publishers, Massachusetts, USA, 1999.
- CHANDRA, V.; ORUGANTI, B.; KUMAR, R. UKDES. **A graphical software tool for the design, analysis e control of discrete**

- event systems.** clue.eng.iastate.edu/~rkumar/PUBS/ukdes.ps. 8:32, 2011.
- CHANDRA, V.; HUANG, Z.; KUMAR, R. **Automated Control Synthesis for an Assembly Line Using Discrete Event System Control Theory.** Transactions on Systems, Man and Cybernetics IEEE Vol. 33, Nº 2, May, 2003.
- COELHO, M. S. **Apostila de Sistemas Supervisórios.** Curso superior de tecnologia em automação e controle de processos industriais contínuos. Instituto federal de educação, ciência e tecnologia de São Paulo campus Cubatão, 2010.
- COSTA, G.O. **Uma Plataforma Computacional de Suporte ao Ciclo de Desenvolvimento de Sistemas Automatizados de Manufatura.** Dissertação (Mestrado em Engenharia de Produção e Sistemas) – Pontifícia Universidade Católica do Paraná, Curitiba, 2005.
- CURY, J. E. R. **Teoria de Controle Supervisório de Sistemas a Eventos Discretos.** V Simpósio Brasileiro de Automação Inteligente, Gramado, 2001.
- DIOGO, R.A.; VICARI, C. A.; LOURES, E. F. R.; Buseti, M. A.; SANTOS, E. A. P. **An Implementation Environment for Automated Manufacturing Systems.** 17th IFAC World Congress. Seoul, Korea, 2008.
- FABIAN, M e HELLGREN A. **Desco – a Tool for Education and Control of Discrete Event Systems.** Kluwer Academic Publishers, 2000.
- FENG, L. e WONHAM, W. **TCT: A computation tool for supervisory control synthesis.** Eighth International Workshop on Discrete Event Systems, Ann Arbor, pp. 388–389, 2006. <http://www.control.toronto.edu/people/profs/wonham/wonham.html>
- GOUYON, D.; PETIN, J. F.; GOUIN, A. **Pragmatic approach for modular control synthesis and implementation.** International Journal of Production Research Vol.42, Nº 14, July, 2004.
- GRIGOROV, L. **Template Design of Discrete-Event Systems.** Project report, School of Computing, Queen’s University, Canada, 2007. <http://www.cs.queensu.ca/~grigorov/>
- GROOVER, M.P. **Automation, Production Systems and Computer Integrated Manufacturing.** Ed. Prentice Hall 2ª ed. New Jersey, 2001.
- IEC 61131-3. **International Electrotechnical Commission Programmable Controllers.** Part 3: Programming Languages, 1998.

- IEEE Standard for SCADA and Automation Systems, IEEE Power Engineering Society, 2008.
- KLINGE, S. **Supervisory control of a manufacturing cell: modeling and implementation**. (Minor Thesis) - Fakultät für Elektrotechnik und Informationstechnik, Otto-von-Guericke-Universität Magdeburg, 2007.
- KRUTZ, Ronald. **Securing SCADA Systems**. Wiley Publishing, Inc. Indiana, 2004.
- CLP ALTUS. Manuais MP399102, MP399103 e MP399104. Disponível em: <http://www.altus.com.br/site_ptbr/>, 01/03/2010.
- MARTINS, V. BREMER, C.F. **Proposta de uma Ferramenta de Integração entre Sistema ERP-SCADA: Caso Prático**. XXII Encontro Nacional de Engenharia de produção. Curitiba-PR, 23 a 25 de outubro de 2002.
- MORAES, C. C.; CASTRUCCI, P. L. **Engenharia de Automação Industrial**. Livros Técnicos e Científicos (LTC), 2001.
- MORAES, C.C.; CASTRUCCI, L.C. **Engenharia de Automação Industrial**, Editora LTC, Brasil, 2008.
- MODBUS. Disponível em: <[http:// http://www.modbus.org/](http://http://www.modbus.org/)>, 05/06/2010.
- PINNHEIRO, J.M.S. Introdução às Redes de Supervisão e Controle. 2006. Disponível em: http://www.projetederedes.com.br/artigos/artigo_redes_de_supervisao_e_controle.php 15/04/2010.
- PENA, P., CURY, J., e LAFORTUNE, S. **Verification of Nonconflict of Supervisors Using Abstractions**. IEEE Transactions on Automatic Control, Vol. 54, No. 12, p. 2803, 2009.
- PENIN, A.R. Sistemas SCADA – 2da. Edición. México D.F, pp. 55-57, 2007
- RAMADGE, P. J. G.; WONHAM, W.M. **The control of discrete event systems**. Proceedings of the IEEE, v. 77, n. 1, pp. 81-98, January 1989;
- REISER, C. **O Ambiente GRAIL para Controle Supervisório de Sistemas a Eventos Discretos: Reestruturação e Implementação de Novos Algoritmos**. Dissertação, Universidade Federal de Santa Catarina, 2005.
- QUEIROZ, M.H. e CURY, J.E.R. **Modular supervisory control of large scale discrete event systems**. Proceedings of the 5th International Workshop on Discrete Event Systems, Ghent, Belgium, pp. 103 – 110, 2000.

- QUEIROZ, M.H. **Controle Supervisório Modular de Sistemas de Grande Porte**. Dissertação (Mestrado em Engenharia Elétrica) - Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, 2000.
- QUEIROZ, M.H. de; CURY, J. E. R. **Controle Supervisório Modular de Sistemas de Manufatura**. SBA. Sociedade Brasileira de Automática, Campinas, v. 13, n. 2, pp. 123-133, 2002.
- QUEIROZ, M.H. de ; CURY, J.E.R. **Synthesis and implementation of local modular supervisory control for a manufacturing cell**. In: Sixth International Workshop on Discrete Event Systems, Zaragoza, 2002.
- QUEIROZ, M.H. **Controle supervisório modular e multitarefa de sistemas compostos**. Tese (Doutorado em Engenharia Elétrica) - Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, 2004.
- REYES, D.A.H. **Las telecomunicaciones aplicadas a los procesos productivos de petróleos mexicanos**. México, 2007.
- RAYMOND, D.R. e WOOD, D. **Grail: A C++ Library for Automata and Expressions**. Journal of Symbolic Computation, 17(4):341–350, 1994.
- ROCHA, Victor. **Automação e Sensoreamento Remoto utilizando Software Livre "SCADA"**. Disponível em: <http://www.vivaolinux.com.br/artigo/Automacao-e-Sensoreamento-Remoto-utilizando-Software-Livre-SCADA/>, 12/05/2011.
- SILVA, Y.G. e QUEIROZ, M.H. **Formal synthesis, simulation and automatic code generation of supervisory control for a manufacturing cell**. Proceedings of the 20th International Congress of Mechanical Engineering, Gramado, Brazil, 2009.
- SILVA, Y.G. **Controle supervisório modular local de sistemas de veículos auto-guiados**. Dissertação (Mestrado em Engenharia de Automação e Sistemas) - Programa de Pós-graduação em Engenharia de Automação e Sistemas, Universidade Federal de Santa Catarina, Florianópolis, 2010.
- SILVA, A.P.G. da e SALVADOR, M. **O que são sistemas Supervisórios?**. Elipse, 2005.
- STOUFFER, K.; FALCO, J. e KENT, K. **Guide to Supervisory Control and Data Acquisition (SCADA) and Industrial Control Systems Security**. Recommendations of the National Institute of Standards and Technology, pp. 1, 2-13, Gaithersburg, 2006.

- SU, R. e WONHAM, W.M. **Supervisor reduction for discrete event systems**, Discrete Event Dynamic Systems, Vol. 14, No. 1, pp. 31-53, 2004.
- VAZ, A. F. e WONHAM, W. M. **On supervisor reduction in discrete-event systems**. International Journal of Control, v. 44, n. 2, pp. 475-491, 1996.
- VIANNA, W.D.S.; BRINGHENTI, P.M. e MARTINS, L.D.S. **Sistema SCADA Supervisório**. Instituto Federal Fluminense de Educação Ciência e Tecnologia. Campos dos Goytacazes, Rio de Janeiro, 2008.
- VIEIRA, A.D.; CURY, J.E.R.; QUEIROZ, M. H. de. **Um modelo para implementação de controle supervisório em controladores lógico programáveis**. In: XVI Congresso Brasileiro de Automação (CBA2006), Salvador, 2006.
- VIEIRA, A. D. **Modelo de implementação do controle de sistemas a eventos discretos com aplicação da Teoria de Controle Supervisório**. Tese (Doutorado em Engenharia Elétrica) - Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, 2007.
- WONHAM, W.M. **Notes on control of discrete-event systems**. Department of Electrical and Computer Engineering, University of Toronto, 2008.