JULIO ARAKAKI

**TÉCNICAS DE DEGENERAÇÃO NO PROJETO DO CONTROLE
DE SISTEMAS PRODUTIVOS**

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

São Paulo

2004

JULIO ARAKAKI

**TÉCNICAS DE DEGENERAÇÃO NO PROJETO DO CONTROLE
DE SISTEMAS PRODUTIVOS**

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

Área de concentração:
Engenharia de Controle e Automação
Mecânica

Orientador:
Prof. Dr. Paulo Eigi Miyagi

São Paulo

2004

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com anuência de seu orientador.

São Paulo, 29 de outubro de 2004.

Assinatura do autor:

Assinatura do orientador:

FICHA CATALOGRÁFICA

Arakaki, Julio

Técnicas de Degeneração no Projeto do Controle de Sistemas Produtivos, São Paulo, 2004. 154p.

Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1. Software dos Sistemas de Controle. 2. Sistemas Produtivos. 3. Degeneração. 4. Sistema Distribuído. 5. Edifícios Inteligentes. I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t.

Ao meu filho Victor e à minha esposa Lílian pela paciência
e compreensão nas minhas ausências.

Aos meus pais (Kaoru e Yolanda) responsáveis pela minha
existência e educação.

Aos meus irmãos (Reginaldo, Beth e Cleuza) e respectivas
famílias, pela união e dedicação em todas as situações.

Enfim a todos os meus familiares.

AGRADECIMENTOS

Ao meu orientador e amigo Prof. Dr. Paulo Eigi Miyagi, pelo auxílio, incentivo e insistência (desde o mestrado).

Ao amigo Prof. Dr. Diolino José dos Santos Filho pelo incentivo, motivações e pelas contribuições.

Ao amigo e irmão Prof. Dr. Reginaldo Arakaki pelas contribuições, pelo apoio e pela presença nos momentos mais difíceis.

Aos professores Dr. Fernando Giorno, Dr. João Maurício Rosário e Dr. Jorge Risco Becerra, participantes da banca, pelas valiosas contribuições.

Aos amigos Arata, Cristina, Emilia, Fabrício e Gladys pelas revisões e contribuições.

A CAAPI Tecnologia (representado pelos amigos Ângelo Frolini, Renato Manzan e outros funcionários) pelo apoio e cobertura nas etapas cruciais.

A todos do Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos da Escola Politécnica da USP e também a todos do Departamento de Ciência da Computação do Centro de Ciência Exatas da PUC-SP que contribuíram, diretamente ou indiretamente, para a conclusão deste trabalho.

RESUMO

A tese explora um conjunto de requisitos especiais de concepção e desenvolvimento de software que asseguram um alto grau de flexibilidade e eficiência para o controle de sistemas produtivos. Desenvolve-se assim, um método que inclui a técnica de degeneração (*redução gradual do nível de serviços de um sistema*) no projeto do software de controle de sistemas produtivos. Apresentam-se inicialmente os conceitos fundamentais considerados no projeto do software de sistemas de controle como: sistemas produtivos, sistemas de controle, sistemas distribuídos, arquitetura em n-camadas (“middleware”) e sistemas multi-agentes. A seguir, introduz-se a aplicação de requisitos padrões para o desenvolvimento do software de controle com qualidade e com a característica de orientação a objetos. O trabalho apresenta também exemplos específicos relacionados com o controle em Edifícios Inteligentes adotados como estudo de casos, que ilustram a aplicação do método desenvolvido. Os respectivos artefatos resultantes da aplicação de cada etapa do método também são descritos e comprovam o potencial desta abordagem.

ABSTRACT

This thesis explores a set of special requirements for software development that assure high degree of flexibility and efficiency for control of productive systems. Thus, it investigates a method that includes the degeneration technique (i.e., a gradual reduction of the service level of a system) in the development of control software for productive systems. The text presents initially the basic concepts considered in the development of control software such as productive systems, distributed control systems, middleware architecture and multi-agent systems. Following, it introduces the application of standard requests for the development of control software with quality and object orientation features. The work also presents specific examples related with the control in Intelligent Buildings which have been adopted as case study and that illustrate the application of the proposed method. The artifacts generated from the application of each step of the method are also described and confirm the potential of the proposed approach.

SUMÁRIO

LISTA DE FIGURAS	V
LISTA DE TABELAS.....	X
LISTA DE ABREVIATURAS	XI
LISTA DE SÍMBOLOS.....	XIII
1. INTRODUÇÃO	1
1.1. Sistema Produtivo e a sua natureza.....	2
1.2. Sistema Produtivo x Sistema Distribuído	3
1.3. Software para Controle de sistemas produtivos.....	6
1.4. Motivações	7
1.4.1. Edifício Inteligente como Sistema Produtivo	8
1.5. Objetivos	10
1.6. Organização do texto	10
2. CONCEITOS FUNDAMENTAIS	12
2.1. Características dos Sistemas Produtivos.....	12
2.2. Sistemas de Controle.....	15
2.2.1. Controle Sequencial	17
2.2.2. Projeto de Sistemas de Controle Sequencial	19
2.3. Sistemas Distribuídos.....	20
2.3.1. Arquitetura baseada em Componentes.....	23
2.4. Orientação a Objetos (OO)	24
2.4.1. Terminologia e Conceitos de OO.....	25
2.4.1.1. Modelagem OO.....	27
2.4.1.2. Padrões de Projeto (“Design Patterns”)	29
2.4.1.3 “Frameworks”	32
2.4.1.4. “Design Pattern” x “Frameworks”	35
2.5. Objetos x Agentes	36
2.5.1. Agentes OO.....	36

2.5.2. Sistemas Multi-Agentes (SMA) e os Sistemas Produtivos	37
2.5.3. Especificação dos agentes num sistema produtivo	41
2.5.4. Arquitetura de SMA tolerante a falhas	44
2.6. Comentários sobre o capítulo.....	45
3. SISTEMAS DE CONTROLE ORIENTADO A DEGENERACÃO	46
3.1. Propriedades do Sistema de Controle	47
3.2. Sistema de Controle Distribuído.....	48
3.3. Reconfiguração em sistemas de controle em tempo real	50
3.4. Sistemas de Controle reconfiguráveis	53
3.4.1. Arquitetura baseada em Componentes.....	53
3.4.1.1. Reconfiguração em sistemas produtivos já existentes	54
3.5. “Middleware” para Reconfiguração	56
3.5.1. Arquitetura de software reconfigurável	57
3.6. Degeração x Regeneração.....	60
3.7. Tratamento de Falhas x Degeração	61
3.8. Requisitos de qualidade de Software.....	68
3.9. Comentários sobre o capítulo.....	70
4. MÉTODO PARA DESENVOLVIMENTO DE SOFTWARE DE SISTEMAS DE CONTROLE COM REQUISITOS DE DEGENERACÃO	71
4.1. Arquitetura de Sistema de Controle com Degeração.....	71
4.2. Projeto do Sistema de Controle com Degeração	74
4.3. Visão geral das etapas para inclusão da degeneração	75
4.4. Descrição das etapas para inclusão da degeneração	77
4.4.1. Etapa1 - Identificação dos pontos críticos do sistema de controle	80
4.4.2. Etapa2 - Modelagem dos mecanismos de degeneração	80
4.4.3. Etapa3 - Especificação técnica do Software Controle de Degeração.....	88
4.5. Comportamento do sistema de controle com os requisitos de degeneração num ambiente de sistema produtivo.....	90
4.6. Comentários sobre o capítulo.....	91

5. ESTUDO DE CASO: EDIFÍCIOS INTELIGENTES	92
5.1. Edifícios Inteligentes e seus subsistemas	93
5.2. Edifícios Inteligentes, seus subsistemas e com requisitos de degeneração	95
5.3. Aplicação do Método Proposto – um Exemplo.....	96
5.3.1. Etapa1 - Identificação dos pontos críticos do sistema de controle	97
5.3.2. Etapa2 - Modelagem dos mecanismos de degeneração	97
5.3.3. Etapa3 - Especificação técnica do Software Controle de Degeneração.....	108
5.3.3.1 Especificação técnica de supervisão de pontos críticos	108
5.3.3.2 Especificação técnica de reconfiguração	109
5.4. Comentários sobre o capítulo.....	110
 6. CONCLUSÕES	 111
6.1. Trabalhos Futuros.....	112
 ANEXO A – NORMA ISO/IEC 9126.....	 113
A.1. Qualidade de produtos de software	113
 ANEXO B - MODELAGEM DE SISTEMAS ORIENTADOS A OBJETOS.....	 115
B.1. Meta modelo UML	115
B.2. Modelo Conceitual da UML	117
B.3. Diagramas UML	120
B.3.1. Diagrama de Casos de Uso (“use case”)	122
B.3.2. Diagrama de Classes	123
B.3.3. Diagrama de Objetos	124
B.3.4. Diagrama de Componentes	124
B.3.5. Diagrama de Distribuição (“deployment”).....	125
B.3.6. Diagrama de Seqüência	125
B.3.7. Diagrama de Colaboração	126
B.3.8. Diagrama de Estados (“Statechart”).....	126

B.3.9. Diagrama de Atividades	126
B.4. MAML (“Multi Agent Modeling Language”)	127
B.5. Perfil UML	128
ANEXO C - EDIFÍCIOS INTELIGENTES	129
C.1. O que é um edifício Inteligente?.....	131
C.2. Edifícios Inteligentes x Sistemas Híbridos	135
C.3. Conforto Ambiental nos Edifícios Inteligentes.....	136
ANEXO D - REDES DE PETRI	139
ANEXO E - METODOLOGIA PFS/MFG	142
REFERÊNCIAS BIBLIOGRÁFICAS	146

LISTA DE FIGURAS

Figura 1.1. Célula de manufatura.....	3
Figura 1.2. Comportamento dinâmico do sistema produtivo da Figura 1.1.....	3
Figura 1.3. Evolução da tecnologia de controle de sistema produtivo (adaptado de LEWIS, 2001).	5
Figura 2.1. Interação entre o sistema produtivo, a Flexibilidade e a Automação (NAKAMOTO [2002]).	13
Figura 2.2. Classificação dos sistemas (adaptado de CASSANDRAS [1993])......	15
Figura 2.3. Controle de malha aberta.	16
Figura 2.4. Controle de malha fechada.	17
Figura 2.5. Arquitetura de um Sistema de Controle (MIYAGI, 1996).	18
Figura 2.6. Etapas para o projeto de sistema de controle seqüencial (MIYAGI (1996)).	19
Figura 2.7. Um sistema distribuído. Note que o nível das aplicações se estende sobre todas as máquinas (unidades de processamento de informações).	21
Figura 2.8. Histórico das linguagens orientadas e não orientadas à objetos (adaptado de OESTEREICH, 1999).	25
Figura 2.9. Principais padrões de projeto (“design pattern”) e seus relacionamentos (adaptado de GAMMA et al., 1994).	32
Figura 2.10. Arquitetura de controle baseado em componentes	35
Figura 2.11. Um sistema complexo (adaptado de JENNINGS e BUSSMAN, 2003).39	
Figura 2.12. Sistema baseado em agentes (adaptado de JENNINGS e BUSSMAN, 2003).	40
Figura 2.13. Sistema produtivo de manufatura: linha de produção flexível (adaptado de JENNINGS e BUSSMAN, 2003).	42
Figura 2.14. Módulo padronizado para uma linha de produção (adaptado de JENNINGS e BUSSMAN, 2003).	42

Figura 3.1. Relacionamento entre os elementos “Model”, “View” e “Controller” no padrão MVC.	48
Figura 3.2. Definição de componentes holônicos numa arquitetura tradicional de um sistema produtivo (adaptado de CHIRN e MCFARLANE, 2000).....	56
Figura 3.3. Camada adicional: “middleware”.	57
Figura 3.4. Exemplo de “Middleware” para reconfiguração.	58
Figura 3.5. Arquitetura de software reconfigurável. Adaptado de WANG e SHIN (2002).	59
Figura 3.6. Estrutura de um componente reutilizável. Adaptado de WANG e SHIN (2002).	59
Figura 3.7. Comparação entre Degeneração e Regeneração.....	61
Figura 3.8. Configuração tolerante a falha. Adaptado de FUJIMOTO e SEKIGUCHI (2003).	62
Figura 3.9. Configuração tolerante a falha (“full duplex”). Adaptado de FUJIMOTO e SEKIGUCHI (2003).	63
Figura 3.10. Rede de Petri representando o método da entrada condicionada.	63
Figura 3.11. Rede de Petri representando o método da rota alternativa.	64
Figura 3.12. Rede de Petri representando o método da recuperação inversa.....	64
Figura 3.13. Subsistema de um sistema produtivo e os componentes de controle, tratamento de falhas e degeneração.	65
Figura 3.14. Rede de Petri representando a integração com a sub-rede de degeneração.	66
Figura 3.15. Subsistema para obtenção de ar refrigerado num Sistema de HVAC. ..	68
Figura 3.16. Obtenção das especificações da arquitetura de software com qualidade.69	
Figura 4.1. Arquitetura típica de um ‘Sistema de Controle’ para sistema produtivo. 72	
Figura 4.2. Arquitetura de um Sistema de Controle para sistema produtivo com a incorporação da degeneração.	73
Figura 4.3. Projeto de sistema de controle com as etapas para inclusão das características de degeneração.....	74
Figura 4.4. Obtenção das especificações da arquitetura de software com qualidade. 76	
Figura 4.5. Geração das especificações com os requisitos de degeneração.....	77

Figura 4.6. Seqüência de etapas para a inclusão dos mecanismos de degeneração...	78
Figura 4.7. Sistema produtivo, seus subsistemas e o controle.	81
Figura 4.8. Sistema Produtivo e seus subsistemas, o subsistema de controle e o reconfigurador (degeneração).....	82
Figura 4.9. Sistema de controle e a Degeneração num subsistema.....	83
Figura 4.10. Acoplamento entre a degeneração e o controle.	84
Figura 4.11. Diagrama de classes dos sensores.....	85
Figura 4.12. Exemplo: modelo MFG para degeneração no caso de ocorrência de falha no sistema de refrigeração.	86
Figura 4.13. Componente para degeneração.	87
Figura 4.14. Componente de controle típico.	88
Figura 4.15. Sistema degenerativo num sistema produtivo.	91
Figura 5.1. Subsistemas em edifícios inteligentes.	94
Figura 5.2. Componente de controle do subsistema <subsis>.	94
Figura 5.3. Subsistemas de edifícios inteligentes com degeneração.....	95
Figura 5.4. Componente de controle associado ao componente de degeneração do subsistema <subsis>.	96
Figura 5.5. Diagrama de classes e, em destaque, os componentes para degeneração para cada subsistema.	98
Figura 5.6. Componente de controle associado ao componente de degeneração do subsistema Transporte.	99
Figura 5.7. Componente de controle associado ao componente de degeneração do subsistema Controle de acesso	99
Figura 5.8. Componente de controle associado ao componente de degeneração do subsistema HVAC.	100
Figura 5.9. Modelo de degeneração para o subsistema Transporte.	100
Figura 5.10. Modelo de degeneração para o subsistema Controle de Acesso.	101
Figura 5.11. Modelo de degeneração para o subsistema HVAC.	101
Figura 5.12. Modelo MFG de conforto térmico com o tratamento de falha (quebra de máquina) do subsistema HVAC (VILLANI et al., 1999).	102

Figura 5.13. Modelo MFG para manter a temperatura desejada para um processo normal.....	103
Figura 5.14. Modelo MFG para degeneração no caso de ocorrência de falha no sistema de refrigeração.	104
Figura 5.15. Modelo MFG para degeneração no caso de ocorrência de falha no acesso controlado.	105
Figura 5.16. Modelo MFG para degeneração no caso de ocorrência de falha no Transporte devido a uma sobrecarga de energia.	106
Figura 5.17. Componente para degeneração para <subsis>.....	107
Figura 5.18. Componente para controle de <subsis>.....	107
Figura 5.19. Componente 'Sensor' composto por 'Sensor digital' e 'Sensor analógico'.	108
Figura B.1. Meta-modelo UML, representando a associação e a generalização (adaptado de FOWLER e SCOTT, 2003).	116
Figura B.2. Histórico de surgimento da UML (adaptado de OESTEREICH, 1999). Evolução das técnicas propostas e respectivos autores.	117
Figura B.3. Relação de dependência.	119
Figura B.4. Relação de associação. O sistema HVAC possui um ou mais sensores.	119
Figura B.5. Relação de Generalização.	119
Figura B.6. Relação de Realização.	120
Figura B.7. Os diagramas da UML.	121
Figura B.8. Diagrama de Caso de Uso (adaptado de BOOCH, 1999).	122
Figura B.9. Diagrama de Classes (adaptado de BOOCH, 1999).	123
Figura B.10. Diagrama de Objetos (adaptado de BOOCH, 1999).	124
Figura B.11. Diagrama de Componentes (adaptado de BOOCH, 1999).	124
Figura A.12. Diagrama de Distribuição (adaptado de BOOCH, 1999).	125
Figura B.13. Diagrama de Seqüência (adaptado de BOOCH, 1999).	125
Figura B.14. Diagrama de Colaboração (adaptado de BOOCH, 1999).	126
Figura B.15. Diagrama de Estados (adaptado de BOOCH, 1999).	126
Figura B.16. Diagrama de Atividades (adaptado de BOOCH, 1999).	127

Figura B.17. Representação hierárquica e as possíveis extensões do padrão UML, através do diagrama de “package”.	128
Figura C.1. Definição de edifício Inteligente.....	134
Figura C.2. Subsistema de Ventilação e Ar condicionado (HVAC).....	138
Figura D.1. Rede de Petri Condição/Evento.	140
Figura E.1. Elementos do PFS.	143
Figura E.2. Elementos do MFG.	143
Figura E.3. Exemplo de um PFS/MFG.	144
Figura E.4. Modelo MFG do sistema produtivo da Figura 1.1.	145

LISTA DE TABELAS

Tabela 2.1. Controle automático (SVC x SED).	17
Tabela 2.2. Dispositivos utilizados no controle seqüencial.	18
Tabela 2.3. Princípios para sistemas de processamento distribuído; adaptado de PUTMAN (2001).	22
Tabela 2.4. Padrões para criação.....	30
Tabela 2.5. Padrões Estruturais.....	30
Tabela 2.6. Padrões Comportamentais.....	31
Tabela 3.1. Tratamento de Falhas x Degeneração	66
Tabela 4.1. As etapas para inclusão da degeneração no método.	79
Tabela 4.2. Exemplos de Pontos Críticos em sistema produtivo.	80
Tabela 4.3.Exemplo: especificação técnica da monitoração de pontos críticos.	89
Tabela 5.1. Alguns Pontos Críticos para o Edifício Inteligente.....	97
Tabela 5.2.Especificação técnica da supervisão de pontos críticos.	109
Tabela 5.3.Especificação técnica da reconfiguração.....	110
Tabela A.1. Características e sub-características para a qualidade de software.	113

LISTA DE ABREVIATURAS E SIGLAS

A/D	- Analógico / Digital
AAA	- Adaptive Agent Architecture
API	- Application Programming Interface
BBS	- Black Board System
C/E	- Condition/Event
CAD	- Computer Aided Design
CORBA	- Common Object Request Broker Architecture
CP	- Controlador Programável (“Programmable Controller”)
CPN	- Colored Petri Net
D/A	- Digital / Analógico
E/S	- Entrada / Saída
EI	- Edifício Inteligente
E-MFG	- Extended Mark Flow Graph
FB	- Function Block
GUI	- Graphics User Interface
HPLC	- Holonic Programmable Logic Controller
HVAC	- Heating, Ventilating, and Air-Conditioning
IA	- Inteligência Artificial
IEC	- International Electrotechnical Commission
ISO	- International Organization for Standardization
MAML	- Multi Agent Modeling Language
MB	- Message Broker
MFC	- Microsoft Foundation Classes
MFG	- Mark Flow Graph
MVC	- Model View Controller
OMG	- Object Management Group
OO	- Orientação a Objetos

OOA - Object Oriented Analysis
 OOD - Object Oriented Design
 OWL - Object Window Library
 P/T - Place/Transition
 PFS - Production Flow Schema
 Pr/T - Predicate/Transition
 SED - Sistemas a Eventos Discretos
 SMA - Sistemas Multi Agentes
 SMA-OO- Sistema Multi Agente Orientado a Objetos
 SPN - Stochastic Petri Net
 SVC - Sistemas a Variáveis Contínuas
 TAO - Taming Agents and Objects
 TPN - Temporized Petri Net
 UML - Unified Modeling Language
 UML-F - Unified Modeling Language – Framework
 UML-RT - Unified Modeling Language – Real Time
 UP - Unified Process

LISTA DE SÍMBOLOS

a_1 e a_2	- Números reais
at	- Atuação
atd	- Atuação com degeneração
co	- Comando do usuário
cod	- Comando do usuário com degeneração
de	- Detecção
ded	- Detecção com degeneração
ei	- Estado interno do sistema
eid	- Estado interno do sistema com degeneração
$g(.)$	- Função de saída resultante
mo	- Monitoração
mod	- Monitoração com degeneração
rec	- Reconfiguração para degeneração
$\langle subsis \rangle$	- Variável para nome do subsistema
$sucd$	- Supervisão de pontos críticos com degeneração
u_1 e u_2	- Vetores de entrada

1. INTRODUÇÃO

A competitividade num mercado globalizado e a necessidade de eficiência na produção incentivam as empresas a investirem em sistemas produtivos mais flexíveis sob diferentes aspectos como volume de produção, tipo de produto e natureza dos recursos envolvidos.

Segundo WADA e OKADA (2002), a flexibilidade significa a facilidade de se modificar os sistemas produtivos de tal forma a melhorar a qualidade dos produtos e/ou serviços e a capacidade de customizar os itens (materiais e/ou informações) a serem produzidos. Neste caso, para a obtenção desta flexibilidade estes pesquisadores indicam que, os sistemas produtivos devem ser baseados em componentes autônomos, descentralizados e distribuídos.

Uma consequência desta necessidade de flexibilidade é o grande interesse em novas tecnologias e arquiteturas para a implementação de sistemas distribuídos aplicados na automação dos sistemas produtivos e cujos softwares de controle são organizados como conjuntos de componentes com alto grau de cooperação (LEWIS, 2001). Esta é uma das principais razões pelas quais os softwares de supervisão e de controle estão tornando-se altamente complexos (SANTOS FILHO, 2000) e invariavelmente possuindo um grande número de módulos com elevado grau de interação. Os módulos, neste caso, são os elementos de hardware e software do sistema. Por exemplo, no contexto de OO (orientação a objetos) estes elementos podem ser entendidos como as classes e no

contexto de SMA (sistemas multi-agentes) estes elementos podem ser entendidos como os agentes.

Neste contexto, esta tese busca explorar um conjunto de requisitos especiais de concepção e desenvolvimento de softwares que conferem a flexibilidade para o controle de sistemas produtivos baseado em técnicas de degeneração, onde o termo degeneração utilizada ao longo deste texto é definido como sendo *uma redução gradual do nível de serviços de um sistema*.

1.1. Sistema Produtivo e a sua natureza

Os sistemas produtivos pertencem a uma classe de sistemas cuja finalidade é a prestação de serviços e/ou a produção de itens (materiais e/ou informações). São sistemas desenvolvidos pelo homem e para o homem (“man made systems”) (ITO, 1991) e cuja evolução dinâmica dos estados é resultante da ocorrência de eventos que são considerados instantâneos. Estes sistemas são caracterizados como sistemas a eventos discretos (SED) (HO, 1989); (MIYAGI, 1996); (SANTOS FILHO, 1998).

Geralmente, os sistemas produtivos são compostos por diferentes elementos, com as características de serem independentes e autônomos, e cujas funcionalidades são essencialmente baseadas na ocorrência de eventos discretos que provocam uma transição abrupta de um estado para outro. Para exemplificar esta característica, a Figura 1.1 apresenta uma célula simplificada de manufatura composta por esteiras (de entrada e de saída), uma máquina de processamento e dois robôs responsáveis pelo transporte de itens na entrada e saída, respectivamente.

Para o exemplo da Figura 1.1 podem-se definir os seguintes eventos discretos: chegada de item, início de processamento, fim de processamento, quebra de máquina, início do descarregamento pelo robô, entre outros, que podem ocorrer no sistema produtivo.

Figura 1.1. Célula de manufatura.

O gráfico da Figura 1.2. ilustra o comportamento dinâmico do sistema produtivo da Figura 1.1.

Figura 1.2. Comportamento dinâmico do sistema produtivo da Figura 1.1.

1.2. Sistema Produtivo x Sistema Distribuído

Segundo MOORE et al. (1999), o aumento da demanda para a utilização dos sistemas de automação cada vez mais complexos (em termos do número de variáveis envolvidas e da intensiva dependência entre elas) torna necessária a adoção de uma arquitetura de controle organizada de modo hierárquica e distribuída integrando funções e

características que permitem a estimativa das tarefas, previsibilidade dos resultados, supervisão, tolerância à falhas, adaptabilidade e aprendizagem.

CAI et al. (2003) afirmam que essa demanda, definida pelos usuários e pelos mercados, determina que os sistemas produtivos sejam também cada vez mais ágeis e flexíveis. Desta maneira, os sistemas de software para o controle dos sistemas produtivos necessitam ser adaptáveis, re-utilizáveis, re-configuráveis e com as características de controle distribuído em tempo real.

Estes requisitos confirmam que a aplicação dos métodos convencionais para o desenvolvimento dos algoritmos de controle pode não ser o suficiente para a obtenção da autonomia e/ou flexibilidade³ desejada. Ou seja, devido à natureza da dinâmica destes sistemas, é difícil, e até mesmo impossível, a aplicação de um modelo analítico (modelo físico e/ou matemático que representa as características de um sistema) que descreva todas as tarefas/operações especificadas para os sistemas produtivos (MOORE et al., 1999). Neste contexto, é necessário o desenvolvimento de novas técnicas e/ou métodos para o projeto de sistemas de controle que atendam às características dos sistemas produtivos.

Outros aspectos a serem considerados são os avanços na tecnologia de redes de comunicação nos ambientes dos sistemas produtivos para a integração e gerenciamento das informações. Por exemplo, a conexão dos dispositivos e/ou equipamentos através de uma tecnologia padronizada como o Fieldbus⁴ possibilita a distribuição de inteligência entre controladores, instrumentos, atuadores e também entre os sensores. Neste caso, todos eles cooperam de forma integrada mas aumentando a complexidade destas interações. A consequência desta interação entre os elementos do sistema é a necessidade de utilização de novas ferramentas e novas tecnologias para a modelagem do comportamento dinâmico destes sistemas assim como para a sua implementação. Assim, o padrão ISO/IEC 61499 (LEWIS, 2001) tem um papel de grande relevância pois foi desenvolvido especificamente para a modelagem de sistemas de controle distribuído, de modo a efetivamente representar a complexidade dos sistemas produtivos.

³ Capacidade do sistema se adaptar a diferentes dinâmicas de seus processos (SANTOS FILHO, 2000).

⁴ Protocolo padronizado para a conexão de dispositivos e equipamentos em ambientes industriais

A Figura 1.3 ilustra a evolução da tecnologia de controle industrial e o conseqüente surgimento dos padrões ISO/IEC 61131-3 (LEWIS, 1996) e ISO/IEC 61499 (LEWIS, 2001). Desde 1950, o crescimento nas funcionalidades proporcionadas pelos sistemas de controle tem sido constante. Os sistemas de controle tornaram-se digitais explorando os recursos dos novos microprocessadores, aumentando a necessidade de se criar regras para reduzir os problemas de integração de dispositivos de diferentes fabricantes (LEWIS, 2001). Em termos de avanços tecnológicos, a Figura 1.3 apresenta como os aspectos de distribuição, integração e independência funcional foram incorporadas ao padrão ISO/IEC 61499, juntamente com a evolução das tecnologias de processamento de informações, de sistemas de comunicação, de modelagem de dados e de Orientação a Objetos (OO).

Figura 1.3. Evolução da tecnologia de controle de sistema produtivo (adaptado de LEWIS, 2001).

1.3. Software para Controle de sistemas produtivos

A disponibilidade de recursos computacionais e o aumento significativo do fluxo de informações possibilitam que os usuários destes sistemas tenham acesso a novas funcionalidades e que estas atendam a especificações cada vez mais rigorosas para os sistemas produtivos. Desta maneira, os softwares de supervisão e controle, para atuarem eficientemente nos ambientes produtivos, devem ser capazes de incorporar constantemente novos requisitos de modo a atender de forma consistente às novas especificações. Conseqüentemente, isto leva ao aumento da complexidade dos componentes do software de supervisão e controle dos sistemas produtivos a serem projetados.

Em relação a este software, o aumento de complexidade da parte que gerencia os processos e das tarefas que fazem a interface com os dispositivos físicos implica numa complexidade maior do código fonte a ser implementado, na inserção de novas características (e/ou novos módulos) e conseqüentemente na introdução de novas interconexões entre estes módulos. Estes softwares devem ainda ser cada vez mais eficientes para suportar o processamento de uma grande quantidade de informações, não se degradando e mantendo sempre um bom funcionamento de forma a atender todos os requisitos determinados pelas necessidades do usuário e do mercado.

Visando, facilitar a manipulação do conjunto de elementos que compõem o software de controle, tem se considerado o uso das técnicas de encapsulamento dos componentes de software através da tecnologia de orientação a objetos (OO) (BOOCH, 1994). Inclusive, os controladores programáveis (CP) (MIYAGI, 1996; MIYAGI et al., 1997b) já dispõem de recursos para adotar esta tecnologia na programação e para a integração sistemática dos CP utilizados nos sistemas produtivos.

Outra conseqüência da complexidade dos sistemas produtivos é a necessidade de investigação da robustez dos softwares para os Sistemas de Controle⁵. Por exemplo, uma característica fundamental para o sistema produtivo é a sua capacidade de se re-

⁵ A partir deste ponto, o termo “controle” é utilizado no sentido amplo, incorporando as atividades de supervisão, comando, monitoração e controle de variáveis.

configurar no caso de ocorrência de alguma situação (evento) não esperada, como uma falha em algum dos seus módulos. Neste caso, é desejável a manutenção das funcionalidades buscando sempre atingir os objetivos traçados inicialmente ou a manutenção de serviços (atividade e/ou operações) essenciais. Entretanto, a grande maioria dos sistemas de controle não possui mecanismos de reconfiguração. Ou seja, em geral, um problema local de um módulo pode afetar todo o funcionamento de um sistema produtivo.

O ideal seria que, no caso de ocorrência deste tipo de situação inesperada, o sistema mantivesse o funcionamento parcial, com degradação mínima no desempenho do restante do sistema. E, havendo a necessidade de desligar todo o sistema, isto deve ser realizado de forma gradual minimizando impactos nos outros módulos e/ou nos usuários do sistema produtivo. Neste caso, é necessária uma supervisão constante de todo o funcionamento do sistema produtivo, sendo que os componentes do software de controle devem possuir um grau adequado de autonomia e de inteligência, assim como os controladores devem estar organizados de modo distribuído e atuando como agentes inteligentes dentro de um sistema integrado (SANZ e ARZÉN, 2003) e cooperativo. Estes requisitos são fundamentais para melhorar a eficiência, flexibilidade e a robustez dos sistemas de controle de sistemas produtivos.

Assim, considera-se essencial que novos métodos para desenvolvimento de softwares que aumentam o grau de confiabilidade dos sistemas de controle no contexto de ocorrência de falhas, sejam investigados. Desta maneira, os conceitos de controle distribuído em tempo real e sistemas baseados em agentes (“multi agent systems” - MAS) também devem ser considerados nas especificações para o projeto de software para controle dos sistemas produtivos.

1.4. Motivações

A maioria dos sistemas de controle dos sistemas produtivos existentes possui algum grau de flexibilidade, robustez e segurança, de tal forma que asseguram o funcionamento destes nos ambientes para os quais foram projetados. Entretanto, em muitos casos, no desenvolvimento tradicional destes softwares de controle, alguns aspectos não são

explicitamente considerados e assim, situações de anormalidade de alguma parte do sistema produtivo podem comprometer o funcionamento de todo o sistema. Desta forma, na implementação de novos sistemas produtivos ou na alteração de sistemas já existentes, é desejável que os softwares responsáveis pelo controle sejam especificados e/ou revisados para que a flexibilidade e a robustez necessárias para o funcionamento efetivo e desejado sejam devidamente e explicitamente incorporadas. Assim, no caso de situações anormais em alguma parte do sistema, um controle inteligente deve conduzir a sua degeneração, desligando os serviços e/ou processos “suavemente”, sem afetar de forma drástica todo funcionamento do sistema produtivo ou ainda se recupere e, de acordo com alguma contingência, retorne totalmente ou parcialmente ao seu funcionamento normal.

1.4.1. Edifício Inteligente como Sistema Produtivo

Um exemplo prático para um sistema produtivo é o chamado Edifício Inteligente (ARKIN e PACIUK, 1995); (ABRAMSON, 1995); (ARAKAKI et al., 1998a); (ARAKAKI et al., 1998b); (MIYAGI et al., 1993); (MIYAGI et al., 1997a); (MIYAGI et al., 1998a); (MIYAGI et al., 1998b). Neste caso, os sistemas de controle são relativamente complexos e envolvem um grande número de módulos que interagem entre si. O conceito de degeneração, que é o tema central deste trabalho, tem papel fundamental para se manter a flexibilidade e a eficiência na obtenção dos resultados (produtos e serviços) com o auxílio dos softwares dos sistemas de controle para este sistema produtivo.

A grande quantidade de subsistemas envolvidos, principalmente para automatizar e integrar os serviços dentro de um edifício, evidencia a necessidade do desenvolvimento de métodos e técnicas que possibilitem a especificação das funções de controle e que auxiliem na implementação de um software com um grau de eficiência suficiente para manter a cooperação entre os subsistemas dos Edifícios Inteligentes.

A integração por sua vez, pode ser realizada através de vários enfoques indicando assim necessidade de se definir uma padronização. Um dos enfoques que pode ser considerado é a integração através de aplicação das técnicas de OO e de Inteligência Artificial (IA)

(RICH e KNIGHT, 1991); (ROWE, 1988); (WINSTON, 1992). Neste caso, a tecnologia OO é adotada como uma forma de estruturar e organizar as informações e a IA é adotada para o gerenciamento e a tomada de decisões.

Num sistema produtivo, o objetivo é a obtenção de um produto final de qualidade, com baixo custo. Além disso, do ponto de vista do processo produtivo, o uso eficiente dos recursos é fundamental para o bom funcionamento dos sistemas produtivos. Num edifício, pode-se considerar como recursos a infra-estrutura, as salas, o ar condicionado, a água, a energia, a acessibilidade, o transporte e, como produto final, tem-se segurança, conforto, mobilidade, higiene, entre outros serviços que dependem do tipo de edifício. O travamento do processo (“deadlock”) pode ocorrer quando não existem recursos suficientes para suprir a necessidade de conforto e segurança ou devido a uma definição/implementação indevida/inadequada do sistema de controle de um prédio.

Os sistemas de controle dos edifícios devem assim serem projetados de tal forma que as situações indesejáveis como os “deadlock” sejam evitados.

A complexidade caracterizada através destes requisitos dos edifícios considerados inteligentes mostra a necessidade de utilização de técnicas e métodos de análise, controle e manutenção que possibilitem o desenvolvimento de softwares de controle mais eficientes que assegurem a flexibilidade dos edifícios.

Como citada em BRENNAN et al. (2001), a complexidade dos processos nestes sistemas (que são caracterizados como sendo distribuídos, concorrentes e estocásticos) e as novas funcionalidades dos dispositivos mecatrônicos associados aos recentes desenvolvimentos de sistemas computacionais motivam a concepção de novos sistemas de controle dos sistemas produtivos compostos por diversos módulos de automação, onde todos interagem de forma cooperativa. Isto envolve a utilização de novas tecnologias de software e hardware que possibilitam o desenvolvimento de softwares de controle dos sistemas produtivos cada vez mais flexíveis incorporando os requisitos de degeneração ou regeneração, ou seja, concebendo sistemas de controle capazes de auto-recuperação no caso das ocorrências de distúrbios não desejados em seu ambiente produtivo.

Portanto, a justificativa para se adotar os Edifícios Inteligentes como estudo de caso, está na necessidade e na possibilidade de se aumentar a eficiência e a flexibilidade dos sistemas de controle através da inclusão dos requisitos de degeneração nos softwares e assim aumentar a inteligência dos Edifícios.

1.5. Objetivos

As motivações citadas induziram para que as investigações relacionadas nesta tese fossem direcionadas para a proposição de um método que sistematiza o projeto do software de controle de sistemas produtivos com a inclusão dos requisitos de degeneração.

Para isso, apresentam-se os conceitos fundamentais relacionados com o projeto de software de sistema de controle e com a adaptação de arquiteturas típicas já existentes de sistemas complexos para suportar os conceitos de degeneração. Para demonstrar a aplicação do método proposto, considera-se como estudo de caso o Edifício Inteligente. Ou seja, o método deverá comprovar seu papel no aprimoramento da capacidade dos subsistemas que compõem os Edifícios Inteligentes de se adaptarem às mudanças das condições do seu ambiente produtivo.

1.6. Organização do texto

A apresentação dos resultados desta investigação segue a seguinte seqüência de capítulos e anexos:

- Capítulo 1: apresentam-se uma introdução ao trabalho, as motivações, os objetivos e a organização do texto.
- Capítulo 2: descrevem-se os tópicos e/ou conceitos que são considerados fundamentais para o desenvolvimento do método proposto. Os itens abordados são: sistemas produtivos, sistemas de controle, sistemas distribuídos, orientação a objetos e outras tecnologias derivadas como “framework” e “design pattern” e sistemas multi-agentes.

- Capítulo 3: são apresentados os requisitos de degeneração nos software de controle para os sistemas produtivos. São apresentadas as técnicas de reconfiguração, tolerância à falhas, arquitetura multi-camadas (“middleware”) e uma análise das arquiteturas para reconfiguração.
- Capítulo 4: apresenta-se o método para desenvolvimento de software para sistemas de controle com a inclusão dos requisitos de degeneração.
- Capítulo 5: apresenta-se o estudo de caso Edifícios Inteligentes e os exemplos que ilustram a aplicação do método proposto.
- Capítulo 6: apresentam-se as principais conclusões do trabalho e as propostas para as futuras pesquisas.
- Anexos A, B, C, D, E: apresentam outros tópicos que são relevantes ao contexto deste trabalho como: a norma ISO/IEC 9126, diagramas UML, Edifícios Inteligentes, Redes de Petri e a metodologia PFS/MFG.

2. CONCEITOS FUNDAMENTAIS

Neste capítulo são apresentados os conceitos fundamentais para o desenvolvimento do método proposto. São apresentadas as definições e características dos sistemas produtivos, sistemas de controle, sistemas distribuídos, algumas técnicas relacionadas com a engenharia de software, como o enfoque de orientação a objetos (OO), e outras tecnologias derivadas como "framework" e "design pattern" que simplificam e flexibilizam o desenvolvimento de artefatos de software e os sistemas baseados em agentes, i.e., sistemas multi-agentes (SMA). O contexto deste trabalho está direcionado para os softwares do sistema de controle dos sistemas produtivos, o foco das discussões está relacionado com as técnicas de engenharia de software que tornam a implementação do software de controle muito mais flexível, de tal maneira que a reconfiguração, a atualização e a adaptação sejam facilitadas. Outros assuntos específicos relacionados com esta investigação são apresentados no Anexo A e no Anexo B.

2.1. Características dos Sistemas Produtivos

Os sistemas produtivos são aqueles sistemas que têm como objetivo a obtenção de um produto que agrega valores como resultado de um processo racional composto por um conjunto de atividades concebidas pelo homem. Um sistema produtivo é composto por um conjunto sequencial de ações e/ou serviços que tem a função de alterar o estado de um item (material ou informações) até a obtenção do produto.

Segundo GROOVER (2000), os sistemas produtivos podem ser considerados flexíveis através da integração dos processos de produção em um sistema altamente automatizado e interconectado por um sistema de manipulação de materiais e, controlado por um sistema distribuído de computadores.

No contexto de sistema produtivo, considera-se neste trabalho, os Edifícios Inteligentes, como o estudo de caso. O produto final oferecido pelos Edifícios Inteligentes é o conforto, a segurança e/ou os serviços. Neste caso, o produto final é dependente do tipo de edifício (hospital, escola ou hotel).

Os sistemas produtivos devem ser flexíveis, de acordo com a necessidade de produção. Ou seja, eles podem produzir simultaneamente um determinado conjunto de produtos de forma eficiente. A flexibilidade dos sistemas produtivos está intimamente relacionada com o nível de automação utilizada (Figura 2.1).

Um sistema produtivo pode ser definido de duas formas (NAKAMOTO, 2002):

- 1) Um agrupamento de múltiplas estações de trabalho automatizadas e interligadas por um sistema automatizado de movimentação e transporte de materiais e ferramentas e/ou informações capazes de realizarem diferentes percursos entre as estações, e controlado por computador;
- 2) Um agrupamento de postos de trabalho de prestação de serviços interligados por uma lógica de movimentação de recurso e itens baseada em processos.

Figura 2.1. Interação entre o sistema produtivo, a Flexibilidade e a Automação (NAKAMOTO [2002]).

A flexibilidade e a automação implicam um sistema produtivo bastante complexo e a conseqüente necessidade de um sistema de controle capaz de gerenciar toda esta complexidade.

Um enfoque tradicional para a solução da complexidade nos sistemas produtivos é a divisão em subsistemas menos complexos e que possuem uma solução de controle mais simples que o sistema como um todo (MOORE et al., 1999).

A Figura 2.2 ilustra uma classificação geral para os sistemas de acordo com CASSANDRAS (1993). Como destacado na Figura 2.2, os sistemas produtivos podem ser caracterizados genericamente como sendo:

- **Sistemas dinâmicos** – os valores das saídas dependem dos valores das entradas e dos valores anteriores das entradas num determinado instante;
- **Invariantes no tempo** – o comportamento não varia ao longo do tempo, ou seja, o comportamento do sistema só se altera se o valor de entrada for modificado;
- **Não lineares** – isto é, seguem a relação $g(a_1u_1 + a_2u_2) \neq a_1g(u_1) + a_2g(u_2)$, onde u_1 e u_2 são dois vetores de entrada, a_1 e a_2 são os coeficientes (que caracterizam um sistema) e $g(.)$ é a função de saída resultante (sinal de saída de acordo com o valor da entrada do sistema);
- **De estado discreto** – as variáveis de estado pertencem ao domínio de inteiros não negativos;
- **Dirigido por eventos** – a ocorrência assíncrona de eventos considerados instantâneos e que causa uma mudança abrupta de estado.

Essas características permitem classificar os sistemas produtivos como sendo Sistemas a Eventos Discretos (SED) (HO [1989]).

Figura 2.2. Classificação dos sistemas (adaptado de CASSANDRAS [1993]).

2.2. Sistemas de Controle

Controlar um sistema produtivo significa supervisionar e corrigir atividades e ações para garantir que elas sejam realizadas conforme planejado. Este processo de controle consiste de três etapas principais que são: 1) medida do desempenho real; 2) comparação do desempenho real com um padrão e; 3) tomada de decisão para corrigir desvios ou padrões inadequados.

O controle pode ser definido como a “aplicação de uma ação pré-planejada para que aquilo que se considera como objeto satisfaça certo objetivo”. Satisfazer certos objetivos, no caso de controle realimentado corresponde a igualar o valor de certa variável física (variável de controle) a um valor de referência. No caso de controle

seqüencial, corresponde a execução de operações conforme uma seqüência pré-estabelecida. (MIYAGI, 1996).

Num processo produtivo qualquer, as condições de operação estão sujeitas às alterações ao longo do tempo derivadas da ocorrência de eventos. Mesmo para dados e/ou informações que são considerados constantes no projeto, na prática, eles podem variar em função das premissas nem sempre possíveis de serem consideradas. Controlar um processo produtivo significa atuar sobre ele, ou sobre as condições a que está sujeito, de modo a atingir algum objetivo - por exemplo, manutenção de um processo sempre próximo de um determinado estado estacionário, mesmo que efeitos externos tentem desviá-lo desta condição. Este estado estacionário pode ter sido escolhido para atender melhor aos requisitos de qualidade e segurança do processo em questão. Num sistema de controle é fundamental ter a clara noção dos objetivos. Ou seja, é inútil influir num processo produtivo sem saber o que se deseja obter.

Existem dois tipos básicos de sistemas de controle: 1) controle de malha aberta (Figura 2.3), onde o sinal de saída é determinado em função do tratamento/processamento do sinal de entrada, não existe uma comparação entre os valores desejados e os efetivamente obtidos e 2) controle de malha fechada (Figura 2.4), onde o sinal de saída é realimentado para a entrada do sistema e através de um regulador de modo que a saída se mantenha sempre próxima de um valor de referência (desejado).

Figura 2.3. Controle de malha aberta.

Figura 2.4. Controle de malha fechada.

A automação se realiza através de um controle automático. Segundo MIYAGI (1996), o controle automático pode ser dividido em duas classes: 1) Sistemas de controle quantitativo, normalmente implementados por controle de sistemas a variáveis contínuas (SVC); 2) Sistemas de controle qualitativo, que podem ser implementados pelo controle de sistemas a eventos discretos (SED). A Tabela 2.1 apresenta uma comparação entre os dois controles.

Tabela 2.1. Controle automático (SVC x SED).

Controle Quantitativo (SVC)	Controle Qualitativo (SED)
Majoritariamente, o objeto de controle manipula informações contínuas.	Majoritariamente, o objeto de controle manipula informações discretas.
Envolve em geral, o controle com realimentação negativa.	Envolve o controle qualitativo e o processamento do comando de controle.
Estrutura de controle é geralmente de malha fechada	Estrutura de controle não é necessariamente em malha fechada.

2.2.1. Controle Seqüencial

É o controle que, baseado numa seqüência pré-estabelecida, ou numa lógica fixa que estabelece uma seqüência, executa passo a passo cada estágio de controle. Nos sistemas de controle seqüencial não existe o conceito de valor de referência, que é substituído pelo comando de execução de uma tarefa. Neste caso, não existe um regulador, e sim um processador de comandos que recebe como entrada o comando (o que deve ser realizado) e o sinal detectado (como o objeto de controle está). Geralmente, estes sinais possuem valores digitais (MIYAGI, 1996).

Os sistemas produtivos considerados neste trabalho envolvem os sistemas de Controle Seqüencial, pois todos os subsistemas são considerados SED, onde a evolução dinâmica

dos estados ocorre de forma paralela e independente. O controle para estes sistemas é caracterizado pelo assincronismo⁶ e paralelismo dos processos envolvidos.

Uma arquitetura básica de um sistema de controle sequencial para os sistemas produtivos é apresentada na Figura 2.5. Neste caso, o sistema de controle interage com o objeto de controle e o operador através dos dispositivos de atuação, detecção, comando e monitoração.

Figura 2.5. Arquitetura de um Sistema de Controle (MIYAGI, 1996).

A Tabela 2.2 apresenta uma classificação e os principais dispositivos utilizados num controle sequencial.

Tabela 2.2. Dispositivos utilizados no controle sequencial.

Função	Dispositivos
Comando	Botoeiras, chaves rotativas, chaves seccionadoras, etc.
Atuação	Válvulas solenóides, contactores, servo-motores, etc.
Detecção	Chaves-limites, potenciômetros, chaves foto-elétricas, termostatos, tacômetros, codificadores, etc.
Monitoração	Lâmpadas, buzinas, alarmes, mostradores, registradores, CRT, etc.
Realização do Controle	Circuitos elétricos, contadores, CP (Controlador Programável), temporizadores, etc.

⁶ Indeterminismo da ocorrência dos eventos em função do tempo.

2.2.2. Projeto de Sistemas de Controle Sequencial

Em MIYAGI (1996) propõe-se uma metodologia para projeto de sistemas de controle sequencial. A Figura 2.6. apresenta as etapas básicas que consistem de análise das necessidades, definição das necessidades, projeto do sistema, projeto do software de controle, implementação do software de controle e os testes.

Figura 2.6. Etapas para o projeto de sistema de controle sequencial (MIYAGI (1996)).

As etapas do projeto de sistema de controle sequencial são definidas da seguinte maneira:

- Etapa 1: *Análise das necessidades* - identificação do objetivo final do sistema; compreensão do objeto de controle, das instalações e dos equipamentos; organização das informações sobre o sistema de controle (dispositivo de controle,

equipamentos de controle, etc); abstração e análise das funções de controle, como os modos de operação e supervisão das instalações e equipamentos.

- Etapa 2: *Definição das necessidades* – definição das funções de controle; definição do fluxo das funções de controle.
- Etapa 3: *Projeto do sistema* – divisão das funções e definição das interfaces; definição e alocação dos sinais de entrada e saída; definição da estrutura do programa.
- Etapa 4: *Projeto do software de controle* – projeto do software relacionado ao modo principal de operação do sistema; projeto de componentes de software relacionados com os modos alternativos de operação.
- Etapa 5: *Implementação do software* – implementação do software de controle através de alguma linguagem específica e sua respectiva instalação nas máquinas.
- Etapa 6: *Testes* – teste de unidade; teste do sistema.
- Etapas 7 e 8: *Operação e manutenção* – operação e correção de funcionamentos inadequados não detectados na etapa de testes.

Num projeto para desenvolvimento de sistema de controle, cerca de 60% a 80% do esforço está relacionado com o desenvolvimento de software, enquanto 20% a 40 % correspondem ao restante. Quando a operação de um sistema de controle é altamente crítica, envolvendo fatores como segurança de seres humanos e um alto custo no caso de alguma falha do sistema, são necessários esforços extras para a validação e verificação do software, antes de sua liberação para produção (HECK et al., 2003).

2.3. Sistemas Distribuídos

Existem diversas definições para sistemas distribuídos na literatura. TANENBAUM (2002) por exemplo, define da seguinte forma: “Sistema Distribuído é uma coleção de entidades que processam informações de modo independente; para os usuários isto deve ser transparente, ou seja, deve parecer um sistema único (“single virtual system” (PUTMAN, 2001)) que atende às suas necessidades”. Esta definição está intimamente

relacionada com a interoperabilidade existente nos sistemas distribuídos. Além disso, existe uma expectativa muito positiva em relação aos sistemas distribuídos, devido, principalmente, aos avanços nos sistemas de comunicação que permitem a rápida transferência de dados entre as entidades envolvidas (HECK et al., 2003).

Em princípio, um sistema distribuído deve ser relativamente fácil de expandir (escalar⁷), devido principalmente à utilização de unidades independentes de processamento de informações. Outra característica importante é a alta disponibilidade, ou seja, os usuários ou aplicações deste tipo de sistema, não devem perceber se alguma parte do sistema está com problemas ou em manutenção. Assim como novos elementos (componentes) podem ser inseridos sem afetar o funcionamento normal.

Os sistemas distribuídos são organizados, normalmente, através de níveis de software, onde, num nível superior estão os usuários e as aplicações e num nível inferior os controladores/sistemas operacionais locais. A Figura 2.7 apresenta uma arquitetura básica de um sistema distribuído. Neste caso, as aplicações se interagem e utilizam todos os recursos disponíveis para realizar o processamento das informações.

Figura 2.7. Um sistema distribuído. Note que o nível das aplicações se estende sobre todas as máquinas (unidades de processamento de informações).

⁷ Aumentar a capacidade de operação (mais recursos) e suporte para mais usuários (por exemplo).

Segundo HECK et al. (2003), as arquiteturas distribuídas são mais robustas do que as arquiteturas centralizadas ou hierárquicas. Por exemplo, na ocorrência de uma falha na Máquina A (da Figura 2.7), o funcionamento das aplicações não deverá ser afetado, pois existem outros recursos (Máquinas) disponibilizados. Além disso, outras máquinas devem atuar no sentido de isolar a falha na máquina A.

PUTMAN (2001) afirma que especificar, projetar e desenvolver um sistema distribuído é complexo. Neste contexto, para a construção de sistemas de processamento distribuído é fundamental que alguns princípios descritos na Tabela 2.3 sejam obedecidos.

Tabela 2.3. Princípios para sistemas de processamento distribuído; adaptado de PUTMAN (2001).

Princípio	Descrição
Interoperabilidade	As entidades de software devem trocar informações mutuamente e de forma correta.
Portabilidade	As entidades de software podem ser utilizadas em diferentes ambientes do sistema de processamento distribuído.
Integração	As entidades de software podem ser integradas de tal forma que para o usuário final seja disponibilizado apenas uma interface de utilização.
Remotabilidade	As entidades de software podem estar distantes uma das outras; neste caso a comunicação pode ser local ou remota.
Concorrência	As entidades podem ser executadas em paralelo.
Heterogeneidade	As entidades de software e os subsistemas podem ser desenvolvidos com diferentes tecnologias. Num sistema distribuído estes elementos devem se comunicar normalmente.
Autonomia	Não existe um controlador central. Vários controladores gerenciam diferentes partes do sistema distribuído.
Evolução	A arquitetura do sistema de processamento distribuído deve considerar mudanças e acomodá-las com um mínimo de impacto.
Escalabilidade	Ao longo do tempo, deve ser capaz de suportar novos usuários, novos processamentos e novos recursos.
Tratamento de falhas	O sistema deve ter a habilidade de se recuperar devido a uma ocorrência de falha.
Qualidade de serviço	O sistema distribuído deve ser capaz de oferecer qualidade nos serviços oferecidos, como disponibilidade, tempo de resposta, etc.
Inexistência de um estado global ou de um “Clock” global	Nos sistemas distribuídos a comunicação e atividades de processamento não são baseadas num “clock” global e único.
Mobilidade	Informações, entidades de software, processamentos e alguns hardwares podem mudar de localização.
Segurança	O sistema deve ter a habilidade de controle de acesso, evitando intrusão e protegendo as informações confidenciais.
Transparência	O sistema de processamento distribuído deve ser como um sistema virtual único (“single virtual system”).

2.3.1. Arquitetura baseada em Componentes⁸

Para o desenvolvimento do software para sistemas de controle complexos (como os sistemas de controle distribuído) é necessária a utilização de ferramentas e tecnologias de software que facilitam o desenvolvimento e a distribuição/instalação (“deployment”) do sistema de controle. As principais propostas neste sentido envolvem: orientação a objetos (OO), padrões de projetos (“design patterns”), “frameworks” e os sistemas multi-agentes (SMA).

Normalmente, os componentes são desenvolvidos de forma mais genérica possível, para serem utilizados em múltiplas e diferentes aplicações, assim como, através de uma combinação coerente, criar outras aplicações. Isto caracteriza a capacidade dos componentes de serem reutilizados em diversas aplicações.

A construção de componentes é baseada na verificação e determinação de padrões genéricos que necessitam de processamentos e soluções computacionais, contidos em diferentes aplicações. O princípio adotado é que o projetista constrói os componentes implementando soluções comuns.

A arquitetura de software de um sistema de controle pode ser baseada em componentes. Por exemplo, considere os componentes em um sistema de controle padrão e hierárquico como: sensores, algoritmos de controle (de alto nível e baixo nível) e atuadores. A maioria destes componentes possui elementos de hardware e software. Similarmente, existem algoritmos de controle padronizados que podem ser utilizados em muitas aplicações (como os controladores PID, filtros de Kalman, controladores baseados em lógica fuzzy e em redes neurais, entre outros). Estes algoritmos padronizados podem ser transformados em componentes genéricos de software e ainda podem ser utilizados em aplicações específicas.

Uma técnica aplicada para o desenvolvimento de componentes é a de orientação a objetos (OO), cujas aplicações estão bastante disseminadas em diferentes áreas, inclusive em sistemas de controle distribuído.

⁸ Uma parte ou um subsistema de um sistema complexo de engenharia (HECK et al., 2003).

2.4. Orientação a Objetos (OO)

Os componentes de software podem ser baseados nos objetos. As características de encapsulamento e invisibilidade de dados (disponíveis através de interfaces específicas), contidos no enfoque de OO, são úteis na construção de componentes reutilizáveis para o projeto de software para os sistemas de controle distribuídos.

Para a implementação de sistemas de software com as características de OO utilizam-se linguagens de programação que incorporam estas características, a Figura 2.8 apresenta o histórico do surgimento do enfoque OO. A Figura mostra a evolução das linguagens desde 1960 (FORTRAN) até 2000 (JAVA e C#) e mostram também quais são as linguagens que possuem as características de OO.

Este enfoque de OO é considerado porque seus conceitos (BOOCH, 1994); (RUMBAUGH et al., 1994) e sua abordagem podem ser utilizados nos sistemas baseados em agentes. Este enfoque é importante para o desenvolvimento de sistemas multi agentes (SMA), onde os agentes são definidos/especificados como entidades (elementos ou pacotes) encapsuladas com suas funcionalidades e seus dados (atributos). Atualmente, em função de suas características positivas, o desenvolvimento de novos sistemas tem sido realizado através de OO. Além disso, a infra-estrutura (hardware e software) que suporta as soluções de computação distribuída, também podem ser construídas com base em objetos (ORFALI e HARKEY, 1998).

Figura 2.8. Histórico das linguagens orientadas e não orientadas à objetos (adaptado de OESTEREICH, 1999).

2.4.1. Terminologia e Conceitos de OO

Neste tópico, são tratados os diversos termos e jargões de OO voltados para a área da Engenharia de Software assim como técnicas de reutilização de projetos (“design pattern”) e de componentização (“frameworks”) comuns para diferentes áreas que consideram os projetos baseados em OO e componentes.

Dependendo da especialização ou do domínio do problema, o significado de cada um destes termos pode aumentar em abrangência ou em especificidade. Alguns afirmam que basta somente a adoção de conceitos como a abstração e o encapsulamento para que

“algo” seja considerado OO. Outros afirmam que é necessário incorporar os conceitos de herança, polimorfismo e persistência para se ter de fato “algo” OO. A seguir serão descritos os principais termos utilizados pelo enfoque de OO:

- **Objeto:** representação de alguma coisa que faz sentido no contexto de uma aplicação. Algo com limites nítidos e significados em relação ao problema. Deve facilitar a compreensão do mundo real e oferecer uma base real concreta para sua implementação.
- **Classe:** representação do encapsulamento de dados e de operações. Uma classe de objetos descreve um grupo de objetos com propriedades semelhantes (atributos), o mesmo comportamento (operações), os mesmos relacionamentos com outros objetos e a mesma semântica.
- **Abstração:** exame seletivo e caracterização de determinados aspectos de um problema. Isola os aspectos que são importantes para algum propósito e suprime os que não são (RUMBAUGH et al., 1994).
- **Atributos:** dados definidos pelos objetos de uma classe.
- **Operações e métodos:** são as funções ou transformações que podem ser aplicadas aos objetos ou classes. Todos os objetos de uma mesma classe compartilham as mesmas operações. A mesma operação pode ser aplicada a muitas classes diferentes. Neste caso, esta operação é dita polimórfica.
- **Encapsulamento:** característica definida quando as informações são armazenadas e protegidas em objetos de uma classe e com acessos restritos.
- **Herança:** obtenção de novas classes a partir das classes já existentes. A classe filha herda os atributos e as funcionalidades da classe pai.
- **Polimorfismo:** indica a habilidade na utilização de diferentes definições para o mesmo nome de método, dependendo do contexto.
- **Persistência:** indica que objetos selecionados e classes de objetos continuam existindo até que eles sejam explicitamente eliminados.

Para o desenvolvimento de um projeto OO (BOOCH, 1994) é necessário o conhecimento e adoção de outros conceitos e termos como:

- **Modelo funcional:** modelo que descreve os aspectos de um sistema relacionados a transformações de valores: funções, mapeamentos, restrições e dependências funcionais. O modelo funcional abrange o que um sistema faz, independentemente de como ou quando é feito.
- **Modelo Dinâmico:** modelo que descreve os aspectos de um sistema relacionados ao tempo e à sequência de operações. O modelo dinâmico incorpora o controle, que é o aspecto de um sistema que define as seqüências de operações que ocorrem, independentemente do que as operações fazem, sobre o que elas atuam ou como são implementadas.
- **Métricas:** medidas de complexidade que permitem a verificação e comparação entre projetos e implementações.

2.4.1.1. Modelagem OO

Para a modelagem de sistemas segundo a abordagem OO, utiliza-se normalmente a UML (“Unified Modeling language”) (BOOCH, 1994); (BOOCH, 1999); (OESTEREICH, 1999); (LARMAN, 2002)

A UML define um conjunto de representações padrões para a documentação do sistema de software. Ela é utilizada como uma forma de visualização de especificações de um projeto, e visa diminuir a distância entre a concepção dos algoritmos e a implementação. Permite a especificação, ou seja, a criação de modelos, de forma precisa e sem ambigüidades, importantes para fundamentar a tomada de decisões relevantes na análise, no projeto e na implementação. Além disso, seus modelos podem ser associados diretamente a uma variedade de linguagens de programação que possuam as características de OO.

A UML permite a documentação da arquitetura do sistema de software e o seu detalhamento e pode ser aplicada em diferentes domínios, inclusive no projeto dos sistemas de controle de sistemas produtivos.

A UML fornece um conjunto de diagramas (Anexo B) que auxiliam na modelagem e no projeto de sistemas de software. Os diagramas são representações gráficas de um conjunto de elementos do sistema de software. A UML inclui nove diagramas:

1. *Diagrama de classes*: diagrama estrutural que mostra o conjunto de classes, interfaces e seus relacionamentos.
2. *Diagrama de objetos*: diagrama estrutural que mostra o conjunto de objetos e seus relacionamentos.
3. *Diagrama de casos de uso* (“*use case*”): diagrama comportamental que mostra o conjunto de “*use cases*” e “*actors*” e seus relacionamentos.
4. *Diagrama de seqüência*: diagrama comportamental que mostra as interações, enfatizando a ordem temporal das mensagens.
5. *Diagrama de colaboração*: diagrama comportamental que mostra as interações, enfatizando a organização estrutural dos objetos que enviam e recebem mensagens.
6. *Diagrama de estados*: diagrama comportamental que mostra a máquina de estado enfatizando os eventos ordenados de acordo com o comportamento dos objetos (visão baseada nos objetos).
7. *Diagrama de atividades*: diagrama comportamental que mostra a máquina de estados do fluxo de atividade para atividade (visão baseada nas atividades a serem realizadas).
8. *Diagrama de componentes*: diagrama estrutural que mostra um conjunto de componentes e seus relacionamentos.
9. *Diagrama de distribuição/instalação* (“*deployment*”): Diagrama estrutural que mostra um conjunto de nós (elementos de infra-estrutura) e seus relacionamentos.

A representação gráfica de cada um destes diagramas é ilustrada através de exemplos no Anexo B. Além disso, nesse anexo, apresenta-se uma extensão da UML para aplicação em projetos baseados em agentes (MAML - “Multi Agent Modeling Language”).

2.4.1.2. Padrões de Projeto (“Design Patterns”)

A grande dificuldade de se construir sistemas de software contendo as características OO e a dificuldade ainda maior para o desenvolvimento de sistemas reutilizáveis levou os pesquisadores, como GAMMA et al. (1994) e LARMAN (2002), a utilizarem as experiências bem sucedidas de implementações práticas para criarem um conjunto de projetos padronizados e abstratos que são comuns em diferentes aplicações. A solução adotada por GAMMA et al. (1994) foi a de expressar os padrões (“patterns”) em termos de objetos e interfaces de tal forma que possibilitem uma solução para um problema dentro de um certo contexto (domínio da aplicação).

Para um melhor entendimento a respeito do “design pattern”, é necessário o conhecimento sobre os elementos básicos que são:

- **Nome (“pattern name”):** utilizado para descrever um problema de projeto, sua solução e conseqüências, em uma ou duas palavras.
- **Problema:** descreve quando aplicar o “pattern”. Neste caso, descreve o problema e seu contexto.
- **Solução:** descreve os elementos que permitem a realização de um projeto, seus relacionamentos, responsabilidades e colaborações. A solução não descreve um projeto ou implementação concreta e particular, pois um “pattern” é como se fosse um “template” que pode ser aplicado em diferentes situações.
- **Conseqüências:** são os resultados e devem permitir uma avaliação crítica das alternativas de projeto e o entendimento dos custos e benefícios da aplicação deste “pattern”.

Os padrões de projeto apresentados em GAMA et al. (1994) podem ser divididos em 3 classes (apresentadas nas Tabelas 2.4, 2.5 e 2.6):

- **Padrões para criação** – Auxiliam a construção de elementos do sistema não dependentes da criação, composição e também da representação de seus objetos. Ou seja, esconde o processo de criação do objeto.
- **Padrões estruturais** – Utilizados para compor estruturas maiores a partir de objetos e classes.
- **Padrões comportamentais** – Representam não somente os padrões de objetos e classes, mas também os padrões de comunicação entre eles.

Tabela 2.4. Padrões para criação.

Padrão	Descrição
“Abstract Factory”	Fornecer uma interface para criação de famílias de objetos relacionados ou dependentes, sem especificar sua classe concreta.
“Builder”	Separa a construção de um objeto complexo de sua representação, de tal forma que o mesmo processo de construção pode ser utilizado para criar diferentes representações.
“Factory Method”	Define uma interface para criar um objeto, mas a subclasse decide de qual classe será instanciada.
“Prototype”	Especifica os tipos de objetos a serem criados através de uma instanciação de protótipo. Cria novos objetos copiando este protótipo.
“Singleton”	Garante que uma classe tenha somente uma instância, e fornece um ponto de acesso global.

Tabela 2.5. Padrões Estruturais.

Padrão	Descrição
“Adapter”	Converte a interface de uma classe em outra interface específica (ou desejada).
“Bridge”	Desacopla a abstração de sua implementação de tal forma que os dois podem variar independentemente.
“Composite”	Compõem os objetos em estruturas em árvore para representar a hierarquia “part-whole”.
“Decorator”	Associa dinamicamente responsabilidades adicionais para um objeto. Fornece uma alternativa flexível para as subclasses estender suas funcionalidades.
“Facade”	Fornecer uma interface unificada para um conjunto de interfaces num subsistema. Define uma interface de alto nível que facilita a utilização do subsistema.
“Flyweight”	Utiliza compartilhamento para suportar um grande número de objetos granularizados de forma eficiente.
“Proxy”	Disponibiliza mecanismo de controle de acesso para outros objetos

Tabela 2.6. Padrões Comportamentais.

Padrão	Descrição
“Chain of Responsibility”	Evita o acoplamento entre quem enviou uma requisição e o receptor desta, através da disponibilização de mais de um objeto para manipular a requisição.
“Command”	Encapsula uma requisição através de um objeto, com possibilidade de parametrização e cancelamento de operações.
“Interpreter”	Dado uma linguagem, define uma representação para sua gramática.
“Iterator”	Fornecer um caminho de acesso aos elementos de um objeto agregado sequencialmente sem expor a sua representação.
“Mediator”	Define um objeto que encapsula a forma que um conjunto de objetos se interage.
“Memento”	Sem a violação de encapsulamento, captura e disponibiliza o estado interno de um objeto de tal forma que o objeto possa ser restaurado para o seu estado anterior.
“Observer”	Define uma dependência entre um-muitos objetos tal que quando um objeto muda o seu estado, todos os seus dependentes são notificados e atualizados automaticamente.
“State”	Possibilita que um objeto altere o seu comportamento quando o seu estado interno é alterado.
“Strategy”	Define uma família de algoritmos, encapsula cada um deles, e os faz intercambiáveis.
“Template Method”	Define um esqueleto de um algoritmo em uma operação.
“Visitor”	Representa uma operação a ser realizada nos elementos da estrutura de um objeto.

Para o desenvolvimento e a implementação do software de controle para sistemas distribuídos com as características de OO, pode-se aplicar os padrões de projetos já existentes. A Figura 2.9 apresenta os possíveis relacionamentos entre os padrões de projeto. Ou seja, no domínio de desenvolvimento de software para os sistemas de controle, a utilização dos padrões de projeto (“design pattern”) tem vantagens principalmente no contexto da implementação, pois se obtém eficiência e redução de custos através do reaproveitamento destes padrões comuns em todos os projetos de software OO e com qualidade.

Figura 2.9. Principais padrões de projeto (“design pattern”) e seus relacionamentos (adaptado de GAMMA et al., 1994).

2.4.1.3 “Frameworks”

Os softwares de controle devem ser capazes de gerenciar a complexidade dos sistemas de grande porte. Neste sentido, devem ser desenvolvidos softwares de controle com

auxílio de tecnologias que facilitem a programação, a integração e principalmente a manutenção dos elementos que o compõem. Assim, foi introduzido o conceito de “frameworks” que são baseados em arquiteturas de software com um domínio de aplicação específico (FAYAD et al., 1999).

Segundo DOUGLASS (2002), cerca de 60% a 90% das novas aplicações de software que são demandadas, são muito similares às aplicações já existentes. Todas estas aplicações que são especificadas necessitam de funções comuns como gerenciamento da criação e destruição de objetos, gerenciamento de memória, planejamento de tarefas, manipulação de interrupções, gerenciamento de filas de eventos para tarefas assíncronas, acessos através de semáforos para tarefas síncronas, execução de máquinas de estado, entre outros. Um “framework” é quase uma aplicação completa que pode ser customizado (especializado) para uma aplicação de uso específico. A diferença entre um “framework” e uma biblioteca é que a biblioteca é passiva (chamada de funções), enquanto o “framework” acoplado a uma aplicação é o responsável pela realização de serviços e/ou atividades específicas. Existem diversos “frameworks” comerciais para aplicações gerais como: Microsoft MFC (Microsoft Foundation Classes) e Borland OWL (Object Window Library) que auxiliam no aumento de produtividade dos desenvolvedores nestes ambientes.

Os “frameworks” são técnicas que induzem uma estrutura hierárquica e viabiliza a reutilização de softwares através de técnicas de OO. FAYAD et al. (1999) citam duas possíveis definições: a primeira diz que “framework” é um projeto reutilizável de todo ou parte de um sistema. É representado por um conjunto de classes abstratas⁹. A segunda definição diz que “framework” é um “esqueleto” de uma aplicação que pode ser customizado pelo desenvolvedor da aplicação. Segundo ele, as duas definições não são conflitantes, pois o primeiro descreve a estrutura de um “framework” enquanto que o segundo descreve o seu propósito. Um “framework” é um projeto reutilizável de um sistema que descreve como o sistema é decomposto em um conjunto de componentes interagindo entre si. Neste caso, o sistema é uma aplicação completa ou algumas vezes

⁹ Em programação OO, uma classe abstrata é aquela no qual não se pode instanciar um objeto. Outras classes podem ser criadas a partir desta (generalização/especialização).

simplesmente um subsistema. Um “framework” descreve os componentes que o compõe e como eles se interagem.

Os sistemas complexos podem assim ser divididos, de forma gradual, em subsistemas até a obtenção de elementos cuja funcionalidade é padronizada através de interfaces coerentes que possibilitam a sua reutilização.

Um “framework” é, neste contexto, um conjunto de classes que se cooperam possibilitando a reutilização de projeto para uma aplicação num domínio específico. Ou seja, um mesmo “framework” pode ser utilizado, por exemplo, para construir editores gráficos em diferentes domínios como desenho artístico, composição musical e CAD mecânico (GAMMA et al., 1994).

Na Figura 2.10 ilustra-se um possível “framework” para um subsistema de segurança de Edifício Inteligente. Neste caso, os elementos são representados por componentes (hardware e software) interconectados através de interfaces de comunicação. Num nível, tem-se, por exemplo, a arquitetura de um sistema de segurança composta pelos componentes de controle, de “interface com o usuário”, de “controle de intrusão” e de “controle de acesso”. No detalhamento do componente de “controle de acesso” incluem-se outros componentes como “controle de catracas”, “controle de portas” entre outros.

A reutilização de componentes pode ser explicitada através do detalhamento do “controle de intrusão”. Neste caso, o componente “câmeras” utilizado no “controle de acesso” também poderá ser utilizado no “controle de intrusão”.

O “framework” representado na Figura 2.10 poderia ainda ser utilizado em outros projetos semelhantes e que necessitam da utilização de sistemas de segurança.

Existem diversos “frameworks” comerciais que são direcionados para uma aplicação específica. Por exemplo, para a telecomunicação, interfaces com o usuário, comércio eletrônico distribuído, sistemas especialistas, entre outros.

Figura 2.10. Arquitetura de controle baseado em componentes

2.4.1.4. “Design Pattern” x “Frameworks”

Segundo GAMA et al. [1994], os “design patterns” e os “frameworks” têm algumas similaridades e diferenças:

- “design patterns” são mais abstratos que os “frameworks” – Os “frameworks” podem ser escritos numa linguagem de programação, utilizados e reutilizados. Enquanto que nos “design patterns” apenas os exemplos podem ser implementados.
- A arquitetura de um “framework” envolve a arquitetura do “design pattern” - Normalmente, um “framework” típico possui vários “design pattern”, mas o inverso nem sempre é verdadeiro.

- “design patterns” são menos especializados que os “frameworks” – “frameworks” sempre possuem um domínio de aplicação particular.

2.5. Objetos x Agentes

Considera-se importante traçar também um paralelo entre os objetos e os agentes, pois estes conceitos estão envolvidos no método a ser estudado. Alguns pesquisadores afirmam que os agentes são objetos com exceção de algumas diferenças. Outros dizem que os agentes e objetos são diferentes, mas compartilham muitas coisas em comum (ODELL, 2002).

Um agente é um encapsulamento de parte de um sistema computacional que está situado em algum ambiente e pode atuar de forma flexível e autônoma neste ambiente buscando atingir os objetivos traçados no desenvolvimento do projeto (JENNINGS, BUSSMAN, 2003).

Um agente é autônomo e faz parte de forma integral do ambiente de processamento. O agente sente o ambiente de forma contínua e realiza alterações de suas atividades de acordo com a necessidade e em conformidade com os objetivos inicialmente definidos. Ele pode ser considerado como um processo inteligente e individual com capacidade para se adaptar e atuar de forma cooperativa com outros agentes. Estes agentes compartilham normalmente dados e informações de controle.

2.5.1. Agentes OO

Como os objetos podem representar qualquer coisa de interesse para um sistema, então, eles podem representar os componentes ou agentes num sistema ou numa aplicação, tais como agente de planejamento, agente de monitoração, agente de degeneração, e assim por diante. Os agentes e seus inter-relacionamentos, e outros componentes de suporte, são os elementos básicos para um sistema multi-agente orientado a objetos (SMA-OO).

Os agentes OO são reutilizáveis quando podem ser utilizados para construir um novo sistema. Isto possibilita a redução de custos de desenvolvimento destes sistemas.

O desenvolvimento baseado em OO atribui mais flexibilidade na estruturação dos agentes, facilita a manutenção e a introdução de novas funções nos sistemas baseados nos agentes.

2.5.2. Sistemas Multi-Agentes (SMA)¹⁰ e os Sistemas Produtivos

Conforme discutido anteriormente existe a necessidade de criação de softwares de controle cada vez mais flexíveis e robustos. A flexibilidade está na possibilidade de inclusão de novas funcionalidades de forma simples e eficiente e a robustez está relacionado com a tolerância a falhas e a capacidade de redução gradual dos serviços (degeneração) no caso de situações críticas (não resolvida no tratamento de falhas). Por outro lado, a utilização de uma plataforma distribuída que interage com sistemas existentes, possibilita a utilização de raciocínio baseado em informações que estão em diferentes níveis de granularidade. A solução distribuída habilita assim a utilização de diferentes paradigmas de soluções de problemas e possibilita a aplicação de critérios de desempenho de forma individualizada (sobre partes do sistema) de modo mais efetivo (JENNINGS e BUSSMAN, 2003)

Os sistemas multi agentes (SMA) estão cada vez mais sendo adotados e aceitos tanto pelas indústrias como pelas instituições de pesquisas, principalmente porque é um paradigma bastante poderoso para o projeto e o desenvolvimento de sistemas de software. Através das implementações práticas dos SMA existe um contínuo aprimoramento que traz, como consequência, o desenvolvimento de novas metodologias, técnicas, linguagens de modelagem, plataformas de desenvolvimento, ferramentas e linguagem de programação que ampliam o potencial deste paradigma (SILVA e LUCENA, 2003). Entretanto, os sistemas baseados em agentes, requerem técnicas adequadas que explorem seus benefícios e suas características peculiares. Assim, os SMA estão cada vez mais sendo utilizados no projeto e na implementação do controle dos sistemas de alta complexidade.

¹⁰ Do inglês “Multi-Agent Systems”.

Em JENNINGS e BUSSMAN [2003] são apresentados alguns benefícios obtidos com a utilização de SMA nos sistemas de controle. São eles: 1) os agentes possibilitam melhores resultados que a implementação centralizada e/ou “standalone”, porque eles levam em conta múltiplos tipos de dados e a integração destes dados de maneira consistente; 2) os agentes são mais robustos porque eles possuem sobreposição de funcionalidades, de tal forma que é possível a obtenção de resultados parciais mesmo com a eventual falha de algum deles; 3) os resultados podem ser obtidos mais rapidamente devido à cooperação entre os agentes; 4) as funcionalidades de diferentes domínios do sistema podem ser alteradas de forma independente, facilitando a sua manutenção; 5) é possível obter uma visão integrada dos resultados de interesse. Além disso, o sistema projetado pode ser aberto de tal forma que novos agentes podem ser inseridos de maneira incremental.

Como citado em SANTOS FILHO (2000), com o aumento da complexidade do sistema de controle, existe uma tendência em adotar-se uma solução baseada na inserção progressiva de restrições ao que era um simples controle de sequenciamento de um processo. Entretanto, este tipo de procedimento resulta em um modelo da solução de controle com considerável dificuldade de interpretação, de manutenção e de capacidade cada vez menor de reutilização, colocando em uma situação delicada todo o projeto de sistemas de controle.

Num sistema complexo (Figura 2.11), o gerenciamento da complexidade pode ser auxiliado através da *decomposição*, onde problemas maiores são quebrados em pequenos problemas; da *abstração*, onde se define um modelo simplificado do sistema em que são enfatizados alguns detalhes e propriedades enquanto outros são suprimidos e da *organização*, no qual se define e se gerencia a inter-relação entre os vários componentes utilizados para solucionar os problemas.

Figura 2.11. Um sistema complexo (adaptado de JENNINGS e BUSSMAN, 2003).

A Figura 2.12 apresenta uma visão de um sistema baseado em agentes. Neste caso, o sistema envolve múltiplos agentes que representam a natureza descentralizada e a necessidade de interações mútuas. Estas interações podem ser simples passagens de informações ou interações baseadas em interação/comunicação tipo cliente-servidor ou interações completas (com mecanismos de cooperação, coordenação e negociação).

Conforme citado anteriormente, o ideal para os processos produtivos é um sistema de controle que seja, ao mesmo tempo, robusto e flexível. Este tipo de sistema é caracterizado como um SED (HO, 1989); (RAMADGE e WHONHAM, 1989) e é concebido pelo homem e para o homem (“man-made systems”) (ITO, 1991). No projeto destes sistemas normalmente utiliza-se um enfoque para o sistema de controle baseado no planejamento global¹¹ e normalmente no contexto de um determinado período de tempo (1 dia, 1 semana, 1 mês, 1 ano, etc.) (JENNINGS e BUSSMAN, 2003). Ou seja, o enfoque é baseado num pré-planejamento e numa visão de controle de malha aberta, onde a criação do plano é desacoplada da execução. Neste caso, se houver a ocorrência de alguma situação de falha (muitas vezes difícil de se prever) ou um evento não

¹¹ O sistema de controle é baseado na especificação antecipada da quantidade total de itens a serem produzidos para um determinado período.

esperado, é necessário o re-planejamento e/ou a re-inicialização (“setup”) de toda a planta, o que prejudica o desempenho global do sistema produtivo.

Figura 2.12. Sistema baseado em agentes (adaptado de JENNINGS e BUSSMAN, 2003).

No contexto dos sistemas produtivos, a flexibilidade pode significar a facilidade com que as linhas de produção podem ser modificadas com o objetivo de melhorar a qualidade dos produtos que estão sendo produzidos ou para a alteração do volume de produção ou a introdução de novos produtos nessas linhas de produção. Normalmente, os sistemas produtivos tradicionais não satisfazem plenamente estes requisitos de flexibilidade e a atividade de gerenciamento é considerada difícil de ser implementada, ineficiente e de alto custo.

2.5.3. Especificação dos agentes num sistema produtivo

Num sistema produtivo, é necessário, inicialmente, especificar os agentes que devem compor um SMA. Para assegurar a robustez e a flexibilidade, os elementos que compõem um sistema produtivo devem ter a capacidade de sobreposição das funcionalidades. Ou seja, deve-se ter, por exemplo, mais de uma máquina com capacidade de realizar uma mesma atividade e/ou operação. Isto é fundamental para a flexibilidade na execução de tarefas em diferentes máquinas no caso de ocorrência de alguma falha ou alguma situação não prevista. Em JENNINGS e BUSSMAN [2003] é apresentada uma implementação de um sistema baseado em multi-agentes, para controle de uma linha de produção. Neste caso, utilizou-se o conceito de sistema produtivo de manufatura modular, onde a linha de produção (Figura 2.13) é composta por módulos padronizados (Figura 2.14). Esta abordagem facilita a manutenção e a implementação do software do sistema de controle.

Neste contexto de modularização, SANTOS FILHO (2000) propõe uma estruturação do processo de modelagem através do “Production Flow Schema” (PFS) (MIYAGI, 1996) e do “Enhanced Mark Flow Graph” (E-MFG) (SANTOS FILHO, 1993), que são técnicas baseadas em redes de Petri (REISIG, 1985); (PETERSON, 1981); (REISIG, 1992); (MURATA, 1989), para definir uma estratégia de controle de modo racional e sistemático. Neste caso, é feito um refinamento dos níveis de abstração através da abordagem “top-down” considerando-se uma concepção modular (atividades) e hierárquica do sistema produtivo.

Os conceitos para modularização e hierarquização auxiliam no desenvolvimento dos sistemas de controle baseado nas abordagens de OO, de componentes e de agentes. Neste caso, os módulos serão a base para a implementação dos elementos (Objetos, Componentes e Agentes) do software de controle dos sistemas produtivos.

Figura 2.13. Sistema produtivo de manufatura: linha de produção flexível (adaptado de JENNINGS e BUSSMAN, 2003).

No exemplo de JENNINGS e BUSSMAN (2003), cada módulo padronizado (Figura 2.14), consiste de uma máquina, três esteiras que movimentam os itens (materiais) num sentido (indicado pelas setas na horizontal), dois “switches” de transporte (indicado pelas setas verticais, que permitem aos itens mudanças de esteira) e uma esteira para alimentação de itens na máquina. O arranjo dos módulos padrões forma uma linha de produção flexível (Figura 2.13). Uma peça pode ser alimentada numa máquina através da esteira inferior ou passar direto pela máquina através da esteira do meio ou retornar para a máquina ou o sistema anterior através da esteira superior.

Figura 2.14. Módulo padronizado para uma linha de produção (adaptado de JENNINGS e BUSSMAN, 2003).

Para controlar este sistema flexível de manufatura, de forma descentralizada, é necessário associar um agente específico para cada elemento (máquina, esteiras, “switches”). Além disso, existe um agente de controle que é responsável por identificar quais operações são necessárias e quais operações já foram realizadas em determinadas máquinas.

Todos estes agentes executam processos paralelamente e não são totalmente independentes pois eles necessitam serem coordenados. Por exemplo, além de selecionar uma máquina para realizar uma operação (seqüência de processos), o agente de controle deve atender outros requisitos como a otimização dos recursos, por exemplo. E, num ambiente produtivo como da Figura 2.13, normalmente existem inúmeros caminhos para os itens envolvidos, mesmo que os objetivos sejam os mesmos. Certamente, os caminhos físicos mais curtos são preferíveis, mas o mais importante é aperfeiçoar o roteamento dos itens, de tal forma a eliminar congestionamentos, evitando uma consequência negativa para o desempenho do sistema. Muitas vezes é necessário definir prioridades para o caso em que o agente de controle identifica mais de uma possibilidade: neste caso ele deve atender a prioridade mais alta. Segundo JENNINGS e BUSSMAN (2003) a avaliação das possíveis soluções para o sistema é realizada usando técnicas de simulação e os dados obtidos, demonstram que o mecanismo baseado em agentes é extremamente robusto em relação a distúrbios e falhas nas unidades de controle.

Considera-se assim que os sistemas de controle baseados em agentes possuem grandes vantagens em relação aos sistemas com estrutura de decisão centralizada. A mais importante é que a solução descentralizada é baseada em tomadas de decisões locais, permitindo um aumento no grau de flexibilidade e na robustez dos sistemas.

No contexto da tese e considerando os edifícios inteligentes (estudo de caso) verifica-se a necessidade de desenvolvimento de agentes de degeneração associados (elevado grau de interação) aos agentes de controle. A arquitetura destes agentes é semelhante para cada subsistema (segurança, transporte, controle de acesso, incêndio, entre outros). Neste caso, as características de distribuição e a aplicação das técnicas de OO e de agentes facilitam o projeto do software de controle para estes sistemas produtivos.

2.5.4. Arquitetura de SMA tolerante a falhas

A capacidade de um sistema de controle de sistema produtivo tratar a ocorrência de falha possibilita uma melhoria significativa na robustez deste sistema. Além disso, devido à característica de distribuição dos SMA e a autonomia e/ou inteligência dos agentes, o desenvolvimento de sistemas tolerante a falhas é facilitada.

Num sistema produtivo, quando uma situação não esperada ocorre, alguns agentes que compõem o software de controle podem ficar indisponíveis devido à falhas como: quebra de máquinas, interrupção de comunicação, entre outras possibilidades relacionadas com o software e o hardware que compõem o sistema produtivo. Assim, outros agentes devem ter a capacidade de re-estabelecer o controle.

Existem diferentes enfoques para o tratamento de falhas em SMA. Alguns destes trabalhos de investigação são descritos a seguir.

No trabalho apresentado por KUMAR e COHEN (2000) propõe-se, a partir das teorias consolidadas na literatura sobre os SMA e a combinação efetiva com os princípios básicos de tolerância à falhas, projetar sistemas robustos baseados nos conceito de agentes. A validação deste enfoque é realizada através de experimentos utilizando uma arquitetura adaptativa baseada em agentes (AAA¹²).

Em KLEIN e DELLAROCAS (1999) é proposta a manipulação de exceções (“exception handling”¹³) para monitorar o progresso global de um sistema multi-agente. Neste caso, os agentes registram um modelo de comportamento normal e, com o tratamento de exceções, que utiliza agentes sentinelas, faz comparações com o comportamento normal e armazena os possíveis erros devido às diferenças. Os agentes que solucionam os problemas interagem com o serviço de tratamento de falhas para detectar e diagnosticar as falhas e ativar ações corretivas.

KAMINKA e TAMBE (1998) utilizam agentes similares que comparam seus próprios estados com os estados de outros agentes para a detecção de possíveis falhas. Neste

¹² “Adaptive Agent Architecture”.

¹³ Em programação, corresponde a uma técnica para tratamento de ocorrências não esperadas.

caso, a técnica de detecção de falhas é inspirada na psicologia social, no qual se utiliza outro agente como fonte de informações.

Assim, verifica-se a possibilidade da utilização de conceitos já existentes sobre os SMA para projetar sistemas com um grau elevado de robustez e autonomia. Com estas características, os SMA podem auxiliar no desenvolvimento de software para os Sistemas de Controle de sistema produtivo com os mecanismos de reconfiguração incorporados.

2.6. Comentários sobre o capítulo

Foram discutidos conceitos de sistemas produtivos, sistemas de controle, sistemas distribuídos, orientação a objetos e sistemas multi-agentes. Todos estes conceitos são importantes para o desenvolvimento deste trabalho, pois a construção dos softwares de controle contendo os mecanismos e conceitos de degeneração necessitam que os elementos que o compõem sejam eficientes e autônomos (Sistemas Multi Agentes) e com características de escalabilidade e/ou manutenção (conceitos de OO, “frameworks” e “design pattern”). Os conceitos apresentados neste capítulo também são importantes para auxiliar a implementação (que não é o escopo desta tese) do software de controle.

3. SISTEMAS DE CONTROLE ORIENTADO A DEGENERAÇÃO

No caso de sistemas produtivos, a degeneração tem um papel fundamental porque nas situações inesperadas e/ou indevidas, como a ocorrência de falhas em geral, não é necessário que todo o sistema tenha que ser imediatamente paralisado. Na maioria dos casos, a capacidade de degeneração permitiria, por exemplo, que o sistema reduzisse gradualmente seu desempenho realizando atividades e ações relacionadas com o processo de detecção e tratamento de falhas. A degeneração, neste contexto, envolve uma reconfiguração do sistema através de um gerenciamento coerente e eficaz realizado pelo software responsável pelo seu controle.

Neste sentido, o software de controle deve adaptar-se de acordo com a mudança da configuração estrutural do sistema e/ou do comportamento dinâmico das aplicações/sistemas devido às ocorrências de situações críticas. Segundo LADDAGA (2000) apud REILLY et al. (2002), um software é adaptativo quando ele evolui e muda seu comportamento à medida que sua evolução indica que ele já não realiza as atividades para o qual foi projetado, ou quando é possível a obtenção de melhores resultados e novas funcionalidades.

Neste capítulo introduzem-se os conceitos que descrevem os requisitos necessários para a realização desta reconfiguração do software de controle nos seus diversos níveis.

3.1. Propriedades do Sistema de Controle

No contexto do presente trabalho, considera-se fundamental que um sistema de controle apresente as seguintes propriedades (citado em WILLS et al. (2001)):

- **Reconfigurabilidade dinâmica / adaptabilidade:** habilidade para reagir rapidamente às mudanças do ambiente sem comprometer a integridade operacional. O sistema com esta propriedade deve suportar o chaveamento “on-line” dos componentes algorítmicos, redirecionando as interconexões e alterando as prioridades dos fluxos de informações.
- **Extensibilidade:** baseada no conceito de “plug-and-play”. O sistema deve ter a capacidade de incorporar as novas técnicas de projetos de algoritmos de controle, novas soluções da tecnologia de sensores e de comunicações assim como aumentar o desempenho das plataformas de hardware na arquitetura do sistema, sem a necessidade de redesenhar os componentes existentes.
- **Interoperabilidade:** o sistema de controle deve ter a capacidade de operar em ambientes distribuídos e heterogêneos.
- **Portabilidade:** os componentes de software devem “rodar” em diferentes processadores, utilizando diferentes linguagens, plataformas de hardware e protocolos de comunicações (normalmente com tempos de respostas compatíveis).
- **Abertura (“openness”):** o sistema com propriedades de reconfigurabilidade e intercambialidade dos seus componentes devem ser concebidas com arquiteturas de software que sejam flexíveis, de tal maneira que suportem ferramentas e algoritmos de diferentes fontes e domínios.

Uma característica importante nos sistemas produtivos atuais é a distribuição dos elementos que o compõem (dispositivos físicos, controladores, software de controle). Ou seja, a capacidade de processamento e o controle de forma distribuída.

3.2. Sistema de Controle Distribuído

As aplicações distribuídas são, normalmente, difíceis de se desenvolver e gerenciar em função das características dinâmicas e de heterogeneidade inerentes à tecnologia de componentes e de protocolos de comunicação (REILLY et al., 2002).

O desenvolvimento de sistemas de controle que possuem as características de adaptabilidade, reutilização e reconfiguração tem motivado a concepção de arquiteturas de software de controle baseadas em componentes distribuídos. No trabalho de CAI et al. (2003), por exemplo, é descrito como se projeta um sistema de controle distribuído baseado numa extensão do padrão MVC (“Model/View/Controller”) (GAMMA, 1994; OESTEREICH, 1999) orientado a objetos e associado à norma ISO/IEC 61499. O MVC é um padrão de projeto (“design pattern”) originado do Smalltalk, que permite a separação da aplicação (“model”) da representação (“view”) e do controlador (“controller”) (Figura 3.1). Neste caso, os elementos do MVC são representados como uma instância de blocos funcionais da norma ISO/IEC 61499, ou seja, neste contexto, os elementos MVC podem ser descritos da seguinte forma:

Figura 3.1. Relacionamento entre os elementos “Model”, “View” e “Controller” no padrão MVC.

- “Model” – blocos funcionais que representam o comportamento lógico num sistema ou o dispositivo que está sendo controlado;
- “View” – blocos funcionais referente à representação visual (“display”) associada com um ou mais “Models”. Corresponde a interface gráfica (“GUI”);

- “Controller” – é um bloco funcional que encapsula as funções de controle a serem realizadas por um ou mais instâncias de “Model”. Neste caso, ele apresenta interfaces apropriadas para os eventos e os dados visando a integração de suas funções com outros blocos de controle.

O padrão internacional ISO/IEC 61499 (LEWIS, 2001) apresenta as diretrizes para a modularização de softwares através de blocos funcionais (FB – “function block”) padronizados que podem ser utilizados no desenvolvimento de sistemas de controle distribuídos para os processos produtivos complexos. Em particular, este padrão trata da construção dos blocos funcionais baseados no padrão ISO/IEC 61131-3 para linguagens de controladores programáveis (CP) (LEWIS, 1996 apud BRENNAN et al., 2002). Assim viabiliza-se a utilização de CP para o desenvolvimento de sistemas de controle distribuído, assegurando as características de independência de implementação. Ou seja, os blocos funcionais podem ser executados e distribuídos em recursos separados. Estes recursos podem ser encapsulados nos dispositivos físicos e possuem controle independente de sua operação. A norma ISO/IEC 61499 define assim uma interface que facilita a troca de informações (mensagens) entre os dispositivos.

O IEC propõe também uma arquitetura baseada em FB como um padrão para os sistemas de controle de processos produtivos industriais, considerando os conceitos de descentralização e de tempo real crítico (“hard real time”, vide item 3.3) (BRENNAN et al., 2002).

Por outro lado, os pesquisadores, em geral, têm investigado o controle de sistemas produtivos com base numa formalização lógica específica para um domínio particular da engenharia. A justificativa desta abordagem está na viabilidade e conveniência de uma base tecnológica padronizada que permita a especificação, projeto, implementação e manutenção dos sistemas de controle dos sistemas produtivos.

Neste contexto, em geral, deve-se assegurar a reconfiguração dos sistemas de controle, utilizando-se componentes em tempo real ou não. Assim, é fundamental que as características de controle em tempo real envolvendo os sistemas operacionais e as arquiteturas dos sistemas de controle estejam bem definidas.

No caso de controle distribuído, consideram-se quais são as funções de controle realizadas através das entidades com características de serem individualizadas, autônomas e que se cooperam. Geralmente, devido a estas características, as entidades são baseadas nos princípios de OO e SMA, citados no capítulo anterior.

A maioria dos trabalhos que se baseiam no enfoque de agentes aplica o conceito de agentes no nível de controle mais elevado como o “planning” e o “scheduling” (JENNINGS e BUSSMANN, 2003). Alguns trabalhos utilizam a tecnologia de agentes no nível mais baixo (nível dos dispositivos) onde, geralmente, o controle é realizado em tempo real. Segundo BRENNAN et al. (2002), a principal barreira de se utilizar SMA no nível de controle em tempo real é o envolvimento dos elementos do ambiente dos sistemas produtivos que são caracterizados por eventos estocásticos. Assim, as restrições de tempo real nem sempre podem ser definidas de forma satisfatória, de tal maneira que se assegure a operação eficiente e segura do sistema.

Entretanto, nos sistemas de controle distribuídos, o controle num nível próximo do sistema físico é fundamental, pois este envolve a coleta de informações e o controle de diversos dispositivos distribuídos num ambiente propenso a diferentes tipos de perturbações.

Um sistema de controle é considerado como um sistema de controle em tempo real se ele possui a capacidade de realizar as tarefas de acordo com as restrições de tempo a ele imposto. Um sistema de tempo real é considerado como um sistema reativo no sentido de que ele reage enviando respostas aos estímulos de entrada vindo do seu ambiente, em intervalos de tempo específicos e bem determinadas. O comportamento de um sistema de tempo real não envolve somente a integridade dos resultados (“correctness”), mas também o tempo necessário para gerar os resultados (“timeliness”).

3.3. Reconfiguração em sistemas de controle em tempo real

Na área da computação o termo tempo real, está relacionado com o funcionamento correto das aplicações não somente baseado nos resultados lógicos computacionais, mas também do tempo nos quais os resultados devem ser produzidos (STANKOVIC, 1988).

Entretanto, os sistemas em tempo real para um sistema produtivo tendem a ser mais complexos, pois devem ter as características de distribuição, inteligência e adaptabilidade num ambiente dinâmico.

As aplicações em tempo real (FARINES et al., 2000) são caracterizadas por restrições que devem ser respeitadas para que se obtenha o comportamento desejado ou necessário. Todas as tarefas¹⁴ em um sistema em tempo real estão sujeitas a um prazo para o seu término (“deadline”). Em princípio, todas as tarefas devem ser concluídas antes do seu “deadline”. Entretanto em função do grau de atendimento do “deadline” pode-se caracterizar dois tipos de tarefas em tempo real:

- **Críticas (“hard real time”)**: quando as tarefas devem ser realizadas num tempo específico de modo que, se completadas após o seu “deadline” têm como consequência, por exemplo, falhas catastróficas no sistema e/ou no seu ambiente. Essas ocorrências podem acarretar em perdas irreversíveis para os equipamentos ou para os seres humanos. Exemplos: sistemas de controle de usina nuclear, sistemas de segurança em aviões, sistemas de navegação espacial.
- **Brandas ou não críticas (“soft real time”)**: quando as tarefas devem ser realizadas num determinado período de tempo de modo que, se completadas após o seu “deadline” acarretam somente numa diminuição no desempenho do sistema. A consequência de um desvio do comportamento normal não implica custos significativos. Exemplos: sistemas de controle de injeção de água numa máquina de lavar, controle de atendimento de elevadores e de pré-aquecimento em edifícios inteligentes, controle de posição de robôs para pinturas em indústrias de automóvel.

Neste contexto, SCARLETT (2003) tem proposto uma investigação sobre como os modelos do padrão ISO/IEC 61499 para controle distribuído em tempo real podem se comunicar com os agentes de um nível superior na hierarquia de controle atendendo as restrições de tempo. Esta interação entre o “mundo” dos agentes e o “mundo” dos dispositivos físicos é baseada nos conceitos fundamentais dos sistemas holônicos. Por

¹⁴ Operações ou atividades realizadas num sistema produtivo.

definição, nos sistemas produtivos holônicos, os “holons” possuem informações de processamento e também da parte física dos dispositivos.

Para a execução e o gerenciamento das operações num sistema em tempo real é necessário um sistema operacional que suporte as características de tempo-real. Do ponto de vista de processamento de dados, existem diversos sistemas operacionais para tempo real, que podem ser utilizados em aplicações gerais. Dentre eles podemos citar: QNX (QNX, 2004), VxWorks, entre outros.

Em FLETCHER e NORRIE (2001), é proposto um sistema operacional em tempo real que permite a execução, o gerenciamento e também a reconfiguração dos blocos funcionais num sistema produtivo. Este sistema operacional é baseado no padrão IEC 61499 com uma arquitetura baseada nos blocos funcionais para execução e gerenciamento em tempo real, uma infra-estrutura para a identificação dos estados destes blocos funcionais e um mecanismo para realizar mudanças na configuração em tempo de execução satisfazendo as restrições de tempo real.

Para introduzir o conceito de degeneração num sistema de controle é necessário que este sistema possua características de reconfigurabilidade.

O objetivo primário é reconfigurar o sistema e obter, como resultado, um sistema ainda flexível, com um comportamento estável e mantendo suas atividades num ambiente de tempo real. Num sistema que utiliza, por exemplo, CP convencionais, a reconfiguração envolve um processo de edição e execução do software de controle durante a própria operação do sistema. A realização/execução destas mudanças (reconfiguração) não é trivial, pois pode ocasionar problemas e deixar o sistema instável, devido ao forte acoplamento existente entre os diversos elementos do software de controle e também da necessidade de manter a sincronização em tempo-real.

Segundo BRENNAN (2002) a reconfiguração de sistema que utilizam CP convencionais envolve o processo de edição em “offline” do software de controle e depois, a colocação em execução no processo. Quando a mudança é realizada, problemas de instabilidade podem surgir devido ao alto grau de acoplamento entre os elementos do software de controle e a inconsistência da sincronização em tempo real. Desta maneira, para se

desenvolver uma metodologia apropriada para minimizar estes problemas, a reconfiguração pode ser analisada em 3 níveis de sofisticação para a implementação:

- **Simples** – quando se utiliza o modelo padrão ISO/IEC 61499 para o acoplamento apropriado dos elementos de software durante a reconfiguração.
- **Dinâmica** – quando se utilizam técnicas apropriadas para uma sincronização correta do software de controle durante a reconfiguração.
- **Inteligente** – quando se explora técnicas de SMA que permitem que o sistema se reconfigure automaticamente em respostas às mudanças ocorridas no ambiente produtivo. Neste caso, a reconfiguração em diversos níveis do controle do sistema produtivo é facilitada pelos agentes de controle.

ZHANG et al. (2000) apresentam neste contexto uma arquitetura baseada em HPLC (“holonic programmable logic controller”)¹⁵ que permite uma reconfiguração multi-nível. A reconfiguração dinâmica é realizada em três níveis: controle de blocos funcionais (alto nível), controle de componentes (nível médio) e controle de tarefas (nível baixo).

3.4. Sistemas de Controle reconfiguráveis

Para suportar a reconfiguração de um sistema é essencial considerar novas arquiteturas do sistema cujas partes envolvem um maior grau de integração. Isso pode ser obtido com a ajuda das tecnologias de comunicação aplicadas para o projeto, integração, operação e manutenção dos softwares de controle para os sistemas distribuídos.

Uma proposta de sistema com a arquitetura reconfigurável, baseada no conceito de sistemas holônicos e em componentes, é apresentada a seguir.

3.4.1. Arquitetura baseada em Componentes

No trabalho apresentado por CHIRN e MCFARLANE (2000) descreve-se uma arquitetura baseada em componentes (na perspectiva de desenvolvimento de software) e

¹⁵ Controlador Programável Holônico para Sistemas de Manufatura Holônicas

nos sistemas produtivos holônicos denominada de HCBA (“holonic component-based architecture”).

A identificação dos componentes é realizada de acordo com os conceitos de projeto “bottom up” e sistemas distribuídos. A identificação das unidades básicas (“holons” e os componentes holônicos) é feita inicialmente de acordo com os elementos fundamentais de uma planta física. Segundo CHIRN e MCFARLANE (2000) os objetos físicos de um sistema produtivo, podem ser caracterizados em dois grupos de acordo com suas propriedades: *recursos* – que realizam as operações produtivas e *item/produto* – que recebem as operações/atividades produtivas (que agregam valor). Desta maneira, definem-se os “*holons*” de *recurso* ou *componentes de recurso* que são os elementos responsáveis pela fabricação, montagem, transporte e testes de itens (materiais e/ou informações). Para a realização destas operações, o componente de recurso possui uma parte responsável pelo controle, tomada de decisão e a troca de informações entre outros componentes. De forma similar, definem-se os “*holons*” de *produto* ou *componentes de item/produto* cuja parte física são as peças, matéria prima, pallets e a parte de controle que contém as informações sobre as rotas, o processo, a tomada de decisões e os dados de produção.

3.4.1.1. Reconfiguração em sistemas produtivos já existentes

A implementação de uma arquitetura que flexibiliza a reconfiguração dos sistemas é dificultada pela impossibilidade de aplicar estes conceitos em ambientes produtivos já existentes. Neste sentido, existem muitas barreiras, principalmente devido ao alto custo para a adaptação de todo o sistema já existente e em operação para adoção destas novas tecnologias. Neste caso, a adaptação de um sistema produtivo já existente (“legacy system”) para uma nova tecnologia é uma atividade dispendiosa em termos de custo e tempo. Por exemplo, num sistema produtivo cuja estrutura hierárquica é composta de máquinas, células e diferentes níveis de produção, existem diferentes funções que são responsáveis pelas atividades produtivas em cada nível de hierarquia do sistema. No nível de máquina (nível mais inferior – “bottom level”) são realizadas operações e controles em tempo real, no nível de célula são realizadas as funções de execução e

supervisão para a coordenação das operações no nível de máquina. E no nível de produção, as funções são mais orientadas à informações, ou seja, as funções de planejamento (“planning”) e a programação das tarefas (“scheduling”) são responsáveis pelas estratégias de controle globais, mas não necessariamente em tempo real. De acordo com as diferentes necessidades de tempo e dados, diferentes dispositivos de comunicação são utilizados em cada um dos níveis. Esta arquitetura leva cada um dos níveis a tornar-se mais centralizado e com uma integração horizontal e conseqüentemente, não compatível com a infra-estrutura baseada em componentes holônicos.

Os componentes holônicos podem ser facilmente identificados e definidos no nível de máquina. Em contrapartida os módulos funcionais de mais alto nível não podem ser associados diretamente a um componente holônico. A proposta apresentada por CHIRN e MCFARLANE (2000) envolve a definição dos componentes holônicos através de módulos pertencentes aos vários níveis (Figura 3.2) de um sistema produtivo. Estes módulos funcionais são decompostos e re-organizados em diferentes componentes holônicos. Desta maneira, esta visão holônica da arquitetura do sistema produtivo, reforça a integração vertical (através dos “holons”), possibilitando uma melhor distribuição física das funções de controle e a integração horizontal (entre os “holons”). Para a integração dos componentes holônicos com o ambiente de software, consideram-se os agentes de comunicação baseados na tecnologia BBS – “black board system” e o MB – “message broker”.

O BBS é utilizado para a comunicação entre os elementos que compõem o “holon”. Ele faz a conexão entre as partes do “holon” que estão situadas nos ambientes de software e nos elementos físicos. O MB é projetado para realizar a comunicação entre os “holons”.

Neste contexto, as aplicações distribuídas baseadas em componentes de software consistem de uma coleção de componentes de software que se comunicam através de um “middleware”. Ou seja, os componentes disponibilizam suas funcionalidades como serviços para outros componentes (REILLY et al., 2002).

Figura 3.2. Definição de componentes holônicos numa arquitetura tradicional de um sistema produtivo (adaptado de CHIRN e MCFARLANE, 2000).

3.5. “Middleware” para Reconfiguração

“Middleware” pode ser uma camada de software utilizado para ocultar a heterogeneidade dos equipamentos / plataformas das aplicações (subsistemas) melhorando, desta maneira, a transparência da distribuição. Em relação à Figura 2.7 do capítulo anterior, que descreve os sistemas distribuídos, acrescenta-se mais uma camada entre as aplicações, que envolve os sistemas operacionais e a rede de comunicação (Figura 3.3). Esta arquitetura possibilita um outro nível de abstração para a interoperabilidade entre os sistemas (hardware e software).

Neste contexto, “middleware” corresponde a um software de conectividade que consiste de um conjunto de serviços disponíveis que possibilita que múltiplos processos, executando em uma ou mais máquinas, interajam através de uma rede.

As aplicações/sistemas existentes (“legacy systems”) podem se beneficiar também dos serviços de “middleware”. Neste caso, estas aplicações são encapsuladas como um conjunto de funções e são acessadas através de um “middleware” que oferece serviços de comunicação para estas funções (BERNSTEIN, 1996).

Figura 3.3. Camada adicional: “middleware”.

A implementação do “middleware” pode ser realizada através do CORBA (“Common Object Request Broker Architecture”) (ORFALI e HARKEY, 1998) da OMG (“Object Management Group”) que disponibiliza uma biblioteca (API - “Application Programming Interface”) orientada a objetos para o gerenciamento do acesso às aplicações remotas (objetos e/ou agentes).

Neste contexto, a reconfiguração responsável pela realização da degeneração pode ser especificada através do conceito de “middleware”. A Figura 3.4 ilustra um exemplo de “middleware” para reconfiguração no qual após uma análise de dependências de todos os componentes do sistema e a partir de estratégias de reconfiguração, altera-se a configuração do componente em análise gerando-se um componente reconfigurado.

3.5.1. Arquitetura de software reconfigurável

Apesar das vantagens de um software de controle com capacidade de reconfigurabilidade existem algumas limitações que dificultam o desenvolvimento de softwares com esta propriedade. WANG e SHIN (2002) citam por exemplo:

- Os aplicativos em geral são projetados e implementados utilizando-se informações proprietárias.

- O comportamento do software de controle é inerente à implementação. Ou seja, em princípio, não é customizável e nem modular.
- A implementação é em geral, específica para uma determinada plataforma.

Figura 3.4. Exemplo de “Middleware” para reconfiguração.

Para solucionar estas dificuldades, WANG e SHIN (2002) propõem uma arquitetura de software (Figura 3.5) baseada numa combinação de modelos orientado a objetos (componentes de software reutilizáveis – software de controle) e uma especificação formal executável (especificações para as aplicações).

Cada componente na proposta de WANG e SHIN (2002) é composto por módulos de software pré-implementados (objetos baseados em modelos OO) e utilizados como blocos (“building blocks”) para a construção do software de controle. Um componente define a funcionalidade de um dispositivo ou subsistema que pode ser o dispositivo de Entrada e Saída (sensor de posição), um algoritmo de controle (controle PID) ou um subsistema mais complexo. A estrutura de um componente de software é composta dos seguintes elementos (Figura 3.6):

Figura 3.5. Arquitetura de software reconfigurável. Adaptado de WANG e SHIN (2002).

Figura 3.6. Estrutura de um componente reutilizável. Adaptado de WANG e SHIN (2002).

- **Interface baseada em eventos** – permite disponibilizar as funcionalidades do componente para o mundo externo. Define as operações que podem ser invocadas de fora do componente.
- **Portas de comunicação** – utilizado para integrar os componentes. São interfaces físicas que possuem mecanismos pelos quais os componentes se interagem. As portas de comunicação possuem atributos que definem o tipo de porta (“send only”, “receive only”, “send-receive”, “buffered” e “non-buffered”), forma de comunicação (“synchronous” e “asynchronous”) e regras de distribuição de mensagens (“first-in first-out”, “priority based”).
- **Lógica de controle (“driver”)** – separa a definição de funções das especificações de lógicas de controle. Suporta a reconfiguração das lógicas de controle no nível de execução. É a parte central para a implementação da reconfiguração após a implementação do comportamento do componente. Esta reconfiguração é baseada no estado corrente do componente e de eventos que chegam pelas portas de comunicação.
- **Protocolos de serviços** – define ambientes de execução ou infra-estrutura para o componente. Possibilitam que os componentes sejam adaptativos para diferentes plataformas.

3.6. Degeneração x Regeneração

Para reforçar o significado de degeneração num sistema de controle de um sistema produtivo, no contexto do presente trabalho, apresenta-se uma comparação com o conceito de regeneração. De forma geral os significados destes dois termos no dicionário da língua portuguesa são:

- “degeneração (do latim *degeneratione*) é a passagem de um estado natural para um estado inferior. Ou seja, alteração para uma situação pior, uma realimentação negativa”.
- “regeneração é o retorno ao modo natural, ou seja, após sofrer modificações, novas alterações são aplicadas para retornar ao modo original”.

Em termos de estados de um sistema produtivo, regeneração pode ser entendida como o retorno ao estado original, após a ocorrência de eventos (automáticos ou não) sobre um estado do sistema não desejado.

Para os sistemas de controle, a regeneração pode ser definida da seguinte forma: na ocorrência de uma situação indesejada e/ou inesperada, o sistema de controle se recupera, mantendo a capacidade de funcionamento com a mesma eficiência e objetivos originais.

A Figura 3.7 ilustra a comparação entre a operação de degeneração e regeneração num contexto de ocorrência de falha no controle de um sistema produtivo. Neste caso, o módulo para tratamento de falhas é ativado. Se a falha for recuperada, o controle é regenerado e o processo normal é restabelecido. Caso contrário, se a falha não for recuperada, o módulo de reconfiguração é ativado de maneira que a seqüência de operações seja modificada para realização de um processo alternativo.

Figura 3.7. Comparação entre Degeneração e Regeneração.

3.7. Tratamento de Falhas x Degeneração

Nos sistemas de controle discretos, o tratamento de falhas permite aumentar a autonomia dos Sistemas Produtivos. Desta maneira, FUJIMOTO e SEKIGUCHI (2003) propõem uma configuração tolerante a falhas baseadas nos controladores programáveis (CP). Neste caso, controladores redundantes são adicionados e eles são ativados quando uma situação¹⁶ anormal é detectada.

A Figura 3.8 apresenta uma configuração de controladores programáveis (CP) e as respectivas interfaces de E/S conectadas numa rede de comunicação e o respectivo CP redundante.

Figura 3.8. Configuração tolerante a falha. Adaptado de FUJIMOTO e SEKIGUCHI (2003).

A Figura 3.9 apresenta uma configuração com controladores redundantes para cada controlador programável.

O processo de recuperação de falhas num sistema produtivo, que possibilita a regeneração e, caso não ocorra a recuperação, possibilita a degeneração, pode ser representado através de técnicas de modelagem baseadas nas Redes de Petri (REISIG, 1985); (MURATA, 1989); (REISIG, 1992). As Redes de Petri possuem diferentes interpretações dependendo do enfoque no qual estão sendo aplicados (Anexo D). Assim, neste item deve ficar claro que está se utilizando as Redes de Petri conhecidas como ordinárias ou Lugar/Transição com arcos inibidoras e habilitadoras.

¹⁶ Estado do CP.

Figura 3.9. Configuração tolerante a falha (“full duplex”). Adaptado de FUJIMOTO e SEKIGUCHI (2003).

No trabalho de RIASCOS (2002) propõe-se uma metodologia baseada em Redes de Petri para o tratamento de falhas em sistemas produtivos onde são citados alguns métodos utilizados para a recuperação de falhas. Nestas representações, onde os macro-eventos são descritas como “atividades” do PFS/MFG (MIYAGI, 1988; SANTOS FILHO, 1993; Anexo E), tem-se:

- **Método da entrada condicionada** (Figura 3.10): após o tratamento da falha, o sistema retorna à operação normal.

Figura 3.10. Rede de Petri representando o método da entrada condicionada.

- **Método da rota alternativa** (Figura 3.11): uma sub-rede (rota alternativa) é utilizada para o tratamento e garantia de recuperação do sistema. Neste caso a “atividade” Q não será realizada após o tratamento da falha

Figura 3.11. Rede de Petri representando o método da rota alternativa.

- **Método da recuperação inversa** (Figura 3.12): a falha é detectada após a execução da “atividade” Q. A sub-rede de tratamento de falha é executada e “atividade” Q é re-executada.

Figura 3.12. Rede de Petri representando o método da recuperação inversa.

A Figura 3.13 ilustra um subsistema de um sistema produtivo com os componentes: ‘sistema de controle’ e de ‘degeneração’. O componente de tratamento de falha está inserido no sistema de controle (para garantir a robustez). O componente de degeneração

é constituído dos componentes de reconfiguração e de supervisão de pontos críticos. A degeneração pode ser vista como um recurso de suporte aos sistemas de controle para superar ocorrências críticas. Ou seja, o componente de degeneração possui um nível de abstração superior ao componente de tratamento de falha.

Figura 3.13. Subsistema de um sistema produtivo e os componentes de controle, tratamento de falhas e degeneração.

Numa representação em Rede de Petri, o processo de degeneração deve ser ativado quando alguma falha não puder ser recuperada (Figura 3.14). Neste caso, não importa o tipo de falha ou o tipo de tratamento de falha, se ela não foi tratada corresponde a uma situação não desejada (e inesperada).

A degeneração é acionada na ocorrência de alguma situação crítica não solucionada. Ou seja, de acordo com a ilustração da Figura 3.14, existe uma supervisão contínua do sistema; se a falha não for resolvida pelos mecanismos (processos) de tratamento de falha, o processo de degeneração é ativado o que resulta na execução de processos alternativos.

Os elementos do “middleware” para reconfiguração podem ser integrados num software de sistema de controle como componentes reutilizáveis, de acordo com a necessidade (WANG e SHIN, 2002).

Figura 3.14. Rede de Petri representando a integração com a sub-rede de degeneração.

Para ilustrar a diferença entre o tratamento de falhas e a degeneração, relacionam-se na Tabela 3.1 falhas que podem ocorrer num sistema produtivo. Neste caso, apresentam-se algumas falhas em subsistemas de um Edifício Inteligente, o respectivo tratamento e o correspondente processo de degeneração.

Tabela 3.1. Tratamento de Falhas x Degeneração

Sub-sistema	Falha	Tratamento de falhas	Degeneração
HVAC	Sensor detecta o ar condicionado não atende ao “set point” especificado.	Procura atuar nas variáveis de controle; Verifica e diagnostica que a água não está fria o suficiente. Liga "chiller" auxiliar; Se não conseguir, comunica a impossibilidade.	É identificado que o tratamento de falha não conseguiu recuperar a operação do ar condicionado; Desliga ou reduz os segmentos menos prioritários; Se não conseguir, acionam sistemas de ventiladores de teto nos pontos que perderam o ar-condicionado.

Transporte	A energia dos elevadores está em níveis críticos	Procurar acionar o sistema de geradores reservas para sustentar o transporte.	Os sistemas de geração não são capazes de sustentar todos os elevadores. Identifica-se como diminuir a carga dos elevadores atuantes e comunica os sistemas de controle. Identifica-se como diminuir o consumo de energia de outras partes do prédio.
Acesso	Os sistemas de eletrônicos falham em conjunto: portas dos elevadores, catracas eletrônicas, ar-condicionado.	Apresenta um desligamento não equilibrado dos elementos de controle.	A monitoração avalia os motivos da falhas, calcula-se a melhor configuração de degeneração e comunica os sistema de controle para: - Desligar alguns elevadores; - Manter todos os sistemas de controle de entrada; - Desligar o ar-condicionado e sustentar a renovação de ar com ventiladores.

A Figura 3.15 mostra o subsistema para a obtenção de ar refrigerado, a partir de resfriamento de ar passando por uma serpentina que é resfriada através da passagem de água fria gerado pelo “chiller”. Este subsistema é correspondente à linha 1 da tabela 3.1. Ou seja, no exemplo da Figura 3.15, se a temperatura no ambiente estiver acima do valor especificado (através do sensor de temperatura), o Tratamento de Falhas tenta corrigir o problema através do acionamento de um “chiller” auxiliar. Se não conseguir resolver o problema, a Degeneração é acionada para desligar segmentos menos prioritários e, neste caso, se for necessário, acionam-se os ventiladores de teto nos ambientes que necessitam a manutenção do resfriamento.

Figura 3.15. Subsistema para obtenção de ar refrigerado num Sistema de HVAC.

3.8. Requisitos de qualidade de Software

Pelo que foi até agora apresentado para o desenvolvimento de um software do sistema de controle é necessário gerar as especificações que atendam às características de OO e de distribuição, citadas nos itens anteriores, assim como os fatores de qualidade de software. A qualidade requerida para o software de controle pode ser obtida obedecendo aos requisitos definidos pela norma ISO/IEC 9126.

Desta maneira, na Figura 3.16 apresentam-se os pontos de vistas considerados no método proposto para a obtenção das especificações da arquitetura do software de controle de um sistema produtivo contendo os requisitos de qualidade (ISO/IEC 9126 (1991) e Anexo A).

Inicialmente especifica-se a arquitetura do software de controle segundo os 5 pontos de vistas (Figura 3.16). Ou seja, é necessário um processo de análise sistemática que consiste de uma avaliação da arquitetura do sistema de controle sob os seguintes pontos de vista:

Figura 3.16. Obtenção das especificações da arquitetura de software com qualidade.

- **Ponto de Vista 1: *Negócio*** – Nesta visão o principal foco de avaliação está no processo de negócio alvo do sistema produtivo para o qual o sistema de controle a ser projetado será aplicado. Como exemplo de processos de negócio num sistema produtivo tem-se: controle de demanda de energia num edifício inteligente, controle de acesso a uma instalação física; controle de combate de incêndios em ambientes construídos; controle do conforto térmico num hospital, controle de sistema distribuído de elevadores, entre outros.
- **Ponto de Vista 2: *Infra-estrutura de Hardware e Software*** – Nesta visão o foco da análise é o conjunto de equipamentos, acessórios, software básico de controle

e outros elementos que constituem a infra-estrutura da solução de controle do sistema produtivo. Ou seja, relaciona-se com os componentes (hardware e software) que compõem o sistema produtivo.

- Ponto de Vista 3: *Segurança* – Nesta visão, o foco está na avaliação da arquitetura do sistema de controle em termos de segurança: quais são as funcionalidades do sistema para manter a integridade dos seus elementos? Como a privacidade é implementada?
- Ponto de Vista 4: *Manutenção* – Nesta visão, avaliam-se as características e os pontos críticos da arquitetura do sistema de controle do ponto de vista de manutenção evolutiva¹⁷ e corretiva¹⁸. Nesta visão é importante lembrar que os sistemas críticos atuais se caracterizam por operações ininterruptas (muitas vezes denominadas “24x7” – vinte e quatro horas por dia, sete dias por semana) onde os ajustes de melhoria e correção devem ocorrer com o mínimo de interrupções.
- Ponto de Vista 5: *Ferramentas de Desenvolvimento* – Nesta visão, avalia-se o ferramental utilizado para desenvolver e manter o sistema de controle identificando as características críticas relacionadas com a operação do sistema.

3.9. Comentários sobre o capítulo

Neste capítulo foram sintetizados os trabalhos que descrevem os principais conceitos de degeneração aplicados aos sistemas de controle dos sistemas produtivos. Foram apresentados diferentes modelos de tratamento de falhas (baseados em Redes de Petri), um modelo com a incorporação de uma sub-rede de degeneração e o conceito de “middleware” aplicado a reconfiguração de sistemas de controle. Estes modelos são considerados no método proposto no próximo capítulo. Apresenta-se também como se analisam as especificações de software considerando a propriedade de qualidade através da norma ISO/IEC 9126 que define as características de um software de qualidade.

¹⁷ Manutenção evolutiva - novas funcionalidades são inseridas na aplicação, gerando-se uma nova versão.

¹⁸ Manutenção corretiva - correções decorrentes de um funcionamento não desejado (mal funcionamento).

4. MÉTODO PARA DESENVOLVIMENTO DE SOFTWARE DE SISTEMAS DE CONTROLE COM REQUISITOS DE DEGENERÇÃO

Este capítulo apresenta um método para determinar os requisitos necessários para a implementação do software de sistema de controle de sistemas produtivos incluindo os requisitos de degeneração. O método é descrito na forma de um encadeamento de atividades. Descrevem-se os artefatos técnicos e/ou produtos gerados de cada uma das etapas na forma de modelos e especificações. Ao final, caracteriza-se o comportamento dinâmico desejado do software para o sistema de controle de sistema produtivo com a degeneração incorporada na sua arquitetura.

4.1. Arquitetura de Sistema de Controle com Degeneração

Uma arquitetura típica de controle de sistemas é apresentada em MIYAGI (1996) (Figura 4.1). Neste caso, para a realização de um controle típico de um sistema produtivo, a atuação sobre o objeto de controle (*at*) é função (*f*) do comando (*co*), da detecção de sinais vindos do objeto de controle (*de*) e do estado interno do sistema (*ei*), ou seja:

$$at = f(co, de, ei)$$

Figura 4.1. Arquitetura típica de um 'Sistema de Controle' para sistema produtivo.

Neste contexto, ilustra-se na Figura 4.2 a inclusão das características de degeneração. O bloco de 'Degeneração' realiza as seguintes atividades:

- 'Supervisão de pontos críticos' - as informações são obtidas pela verificação contínua dos pontos considerados vitais no sistema produtivo.
- 'Reconfiguração' - a reconfiguração é realizada a partir das informações de pontos críticos obtidos pela atividade de 'Supervisão de pontos críticos'.

Assim, a atuação com degeneração (*atd*) é função (*fd*) do comando com degeneração (*cod*), da detecção com degeneração (*ded*) e do estado interno do sistema com degeneração (*eid_i*), ou seja:

$$atd = fd(cod, ded, eid_i).$$

O próximo estado interno com degeneração (*eid_{i+1}*) é função (*g*) da reconfiguração (*rec*) e do estado interno: $eid_{i+1} = g(rec, eid_i)$ e a reconfiguração para a degeneração (*rec*) é função (*h*) dos pontos críticos (*pc*) e da supervisão destes pontos críticos (*sucd*):

$$rec = h(sucd, pc)$$

Figura 4.2. Arquitetura de um Sistema de Controle para sistema produtivo com a incorporação da degeneração.

Os mecanismos para a degeneração do sistema de controle atuam da seguinte forma: primeiro detecta-se a situação de falha ou de anormalidade, através da identificação dos *pontos vitais*¹⁹ do sistema de controle; avalia-se o contexto segundo critérios de prioridades estabelecidas de acordo com os pontos críticos (estado crítico do sistema). Finalmente acionam-se os mecanismos automáticos ou com intervenção humana para reconfigurar o sistema de controle para operar de maneira degradada conforme a programação estabelecida. A situação de falha fica assim sob constante acompanhamento pelos componentes de tratamento de falhas (inserido no sistema de

¹⁹ Partes do sistema com grandes probabilidades de ocorrerem situações críticas.

controle) e o de degeneração, até que se restabeleça a condição de regeneração (retorno ao funcionamento normal).

4.2. Projeto do Sistema de Controle com Degeneração

Em relação às atividades de projeto de um sistema de controle típico apresentado em MIYAGI (1996), o método proposto neste trabalho inclui novas etapas ao projeto de sistema de controle. Estas novas etapas inserem as características de degeneração.

Figura 4.3. Projeto de sistema de controle com as etapas para inclusão das características de degeneração.

Para este novo método, todas as etapas devem ser aplicadas levando-se em conta as características de distribuição e de degeneração. Neste caso, para o projeto de sistema de controle (Figura 4.3) existem duas situações possíveis:

1. *Projeto novo*: desenvolvimento de um novo projeto do sistema de controle. Neste caso, todas as etapas deverão levar em conta as características de distribuição e a inclusão dos requisitos de degeneração.
2. *Projeto já existente*: deve-se adaptar o sistema de controle existente. Neste caso, podem-se utilizar os conceitos para a definição de componentes a partir do sistema já existente (“legacy systems”, apresentados no item 3.4.1.1 do capítulo 3) e posteriormente, aplicar as etapas descritas no item 4.3.

4.3. Visão geral das etapas para inclusão da degeneração

O método proposto consiste em identificar, modelar e especificar os mecanismos de degeneração através de um processo de análise sistemática da especificação da arquitetura do sistema de controle. Considera-se, para a análise da arquitetura de software do sistema de controle com as características de qualidade (Figura 3.16 do capítulo 3) a inclusão dos requisitos para controle, ou seja, deve-se analisar o sistema sob um novo ponto de vista (Figura 4.4):

- Ponto de Vista 6: *Requisitos de Controle* – nesta visão, a análise é feita baseada nas características do sistema de controle como tipo, dinâmica e os algoritmos de controle.

O destaque (pontilhado na Figura 4.4) indica que o ‘Ponto de vista 6’ (*Requisitos de controle*) poderia estar associado ao ‘Ponto de vista de 1’ (*Negócio*). Esta separação tem como objetivo reforçar a inclusão dos requisitos de degeneração.

Os outros pontos de vistas (1, 2, 3, 4, 5) necessários para os requisitos de qualidade da especificação da arquitetura de software, são descritos no item 3.8 do capítulo 3.

Figura 4.4. Obtenção das especificações da arquitetura de software com qualidade.

A seguir apresenta-se a estrutura básica da proposta para a obtenção das especificações do software de controle com os requisitos necessários para a degeneração. Neste caso, as especificações da arquitetura do sistema de controle são as entradas para o processo que gera as especificações do respectivo software de controle, considerando os requisitos de degeneração.

A Figura 4.5 apresenta o contexto para aplicação das etapas para geração dos requisitos de degeneração sobre as especificações da arquitetura do software do sistema de

controle. Os resultados são as especificações da arquitetura do software do sistema de controle com os requisitos de qualidade e degeneração inseridas.

Figura 4.5. Geração das especificações com os requisitos de degeneração.

Com as especificações obtidas, pode-se realizar a implementação do software do sistema de controle onde se tem incorporado os requisitos responsáveis pela degeneração no caso de ocorrência de uma situação crítica.

4.4. Descrição das etapas para inclusão da degeneração

Na Figura 4.6 ilustra-se a seqüência das etapas propostas, através do detalhamento do bloco '*Etapas para geração dos Requisitos de Degeneração*' (Figura 4.5). Inicialmente identificam-se todos os pontos críticos onde determinadas ocorrências inesperadas poderiam afetar o funcionamento normal do sistema produtivo. Desenvolve-se, a partir dos pontos críticos identificados, a modelagem dos requisitos de degeneração e a identificação dos componentes que devem ser utilizados na implementação e posteriormente na revisão e ajustes finais destes mecanismos de degeneração.

Figura 4.6. Seqüência de etapas para a inclusão dos mecanismos de degeneração.

A Tabela 4.1 apresenta uma descrição mais detalhada das etapas propostas. A Tabela apresenta uma coluna com as atividades a serem realizadas em cada etapa, uma coluna com os resultados (produtos) obtidos pela realização das atividades e uma coluna de referência com indicação da tecnologia considerada em cada etapa. Nos itens que se seguem, apresenta-se a descrição mais detalhada de cada etapa proposta.

Tabela 4.1. As etapas para inclusão da degeneração no método.

ETAPAS	ATIVIDADES DA ETAPA	PRODUTOS RESULTANTES	REFERÊNCIAS CONSIDERADAS
Etapa 1 Identificação dos Pontos Críticos	<ul style="list-style-type: none"> • Mapeamento dos pontos críticos com as degenerações 	<ul style="list-style-type: none"> • Relação de Degeneração do Sistema 	<ul style="list-style-type: none"> • ISO 9126 • Padrão de Qualidade estabelecido para o sistema de controle
Etapa 2 Modelagem dos Mecanismos de Degeneração	<ul style="list-style-type: none"> • Modelagem Estática • Modelagem Comportamental • Identificação de Componentes de Supervisão • Identificação de Componentes de Reconfiguração (Degeneração) 	<ul style="list-style-type: none"> • Modelagem com a degeneração incorporada • Especificação Preliminar dos componentes) 	<ul style="list-style-type: none"> • Padrões e Templates de modelagem
Etapa 3 Especificação técnica do Software do Sistema de Controle de Degeneração	<ul style="list-style-type: none"> • Especificação da Arquitetura com Degeneração • Especificação dos Componentes de Supervisão • Especificação dos Componentes de Reconfiguração • Especificação de outros componentes auxiliares 	<ul style="list-style-type: none"> • Especificações técnicas dos componentes 	<ul style="list-style-type: none"> • Padrões e Templates de especificações
Implementação incluindo a degeneração	<ul style="list-style-type: none"> • Implementação do sistema de controle com degeneração 	<ul style="list-style-type: none"> • “middleware” de reconfiguração 	
Revisão e Ajustes dos Mecanismos de degeneração	<ul style="list-style-type: none"> • Elaboração de Planilhas de Acompanhamento da operação do sistema • Testes incluindo a Degeneração e avaliação dos Resultados 	<ul style="list-style-type: none"> • Planilha de Acompanhamento • Registros de Testes • Relação dos Ajustes realizados 	<ul style="list-style-type: none"> • Padrões e Templates • Critérios e Padrões de Ajustes

4.4.1. Etapa1 - Identificação dos pontos críticos do sistema de controle

As atividades relacionadas nesta etapa determinam o que pode ser degenerado, quando o sistema atinge uma situação crítica em seu desempenho. Para isso, identificam-se os pontos críticos, a partir da avaliação do sistema. Com base na lista de pontos críticos do sistema, define-se o que pode ser degenerado mantendo o sistema em operação com todas as funcionalidades possíveis e disponíveis no momento.

O artefato fundamental a ser gerado nesta etapa é o mapeamento dos pontos críticos identificados e das possíveis degenerações em caso de ocorrências de falhas ou de eventos críticos e inesperados. A Tabela 4.2 apresenta uma relação de alguns exemplos de pontos críticos no sistema produtivo Edifício Inteligente e as respectivas degenerações.

Tabela 4.2. Exemplos de Pontos Críticos em sistema produtivo.

Ponto Crítico	Degeneração
Fluxo de água fria num sistema de Ventilação e Ar-condicionado. (pode falhar)	<p>É identificado que o tratamento de falha não conseguiu recuperar a operação do ar condicionado;</p> <p>Desliga ou reduz os segmentos menos prioritários;</p> <p>Se não conseguir, aciona sistema de ventiladores de teto nos pontos que perderam o ar-condicionado</p>
Controle de acesso em um ambiente de acesso seguro. (pode falhar).	Liberar o acesso aos técnicos de manutenção, ou ativar o acesso através de uma entrada especial, pois o serviço não pode parar.
Sistemas eletrônicos de portas dos elevadores e escadas rolantes. (sobrecarga de energia)	<p>Desligar alguns elevadores;</p> <p>Manter portas de escadas destravadas.</p>

4.4.2. Etapa2 - Modelagem dos mecanismos de degeneração

Com base na relação de degeneração estabelecida na Etapa 1 realiza-se a modelagem da estrutura e do comportamento da degeneração. Isto deve ser feito para todos os

subsistemas que compõem o sistema produtivo. A Figura 4.7 ilustra um exemplo com vários subsistemas que compõem um sistema produtivo. Neste caso, as setas indicam as interações entre os subsistemas (sistema produtivo distribuído).

Figura 4.7. Sistema produtivo, seus subsistemas e o controle.

Com a incorporação da degeneração os subsistemas do sistema produtivo devem interagir com o módulo 'Reconfigurador' que tem a responsabilidade de configurar estes subsistemas de tal forma que o subsistema de controle possa conduzir a degeneração. Incluindo-se o subsistema de reconfiguração e o seu fluxo de informações ("middleware" de reconfiguração), obtém-se uma arquitetura representada pela Figura 4.8.

Figura 4.8. Sistema Produtivo e seus subsistemas, o subsistema de controle e o reconfigurador (degeneração).

A Figura 4.9 ilustra o detalhamento de cada subsistema em termos de componentes para reconfiguração. Ou seja, cada subsistema possui o 'Componente de controle' e o 'Componente de degeneração'.

Figura 4.9. Sistema de controle e a Degeneração num subsistema.

Todos os subsistemas que compõem o sistema produtivo devem ser modelados com as características de degeneração incorporados ao modelo. Desta forma, os artefatos técnicos produzidos nesta etapa são:

1) Modelos

Modelagem Estática: modelo estrutural dos elementos do sistema de controle, juntamente com os elementos da degeneração. Esses elementos se caracterizam por permitir o comando, a monitoração, a decisão, a detecção e a atuação sobre os elementos da arquitetura do sistema. Esse modelo é baseado em diagramas de classes da abordagem OO, onde se aplicam técnicas de relacionamentos entre as classes como: herança, composição, associação.

Para incluir o mecanismo de degeneração é necessário que o reconfigurador e a supervisão dos pontos vitais do sistema sejam integrados ao controle.

Assim, a Figura 4.10 ilustra esta integração. A célula de degeneração é um elemento especial, acoplado aos elementos de controle. A degeneração interage sobre os

elementos de controles que consideram combinações e seqüências lógicas dos sinais de sensores e atuadores dos sistemas de controle típicos. As principais classes são:

- Célula de Controle, Módulo Sinais Sensoriais e Módulo Sinais para Atuadores - constituem as classes de controle típico;
- Célula de Degeneração, Supervisor e Reconfigurador - constituem as classes de degeneração, apoiadas pelas células de controle.

Figura 4.10. Acoplamento entre a degeneração e o controle.

Assim, a supervisão de um ponto vital acontece diretamente sobre os elementos de controle para identificação de situações críticas. Os reconfiguradores atuam sobre os elementos de controle para programar e flexibilizar (degenerar) as funcionalidades de controle do sistema produtivo.

A Figura 4.11 apresenta como exemplo o detalhamento do componente 'Sensor'. Neste exemplo através do mecanismo de herança (BOOCH, 1999), dois tipos de sensores podem ser modelados: sensores digitais e sensores analógicos. Neste exemplo, a representação de um sensor de temperatura pode possuir um atributo, por exemplo,

tempRef que indica uma temperatura de referência e o método *getTempRef*, que possibilita a leitura desta informação. Estes sensores são os responsáveis pela determinação do estado crítico.

Figura 4.11. Diagrama de classes dos sensores.

Modelagem Comportamental: modelo do comportamento dinâmico dos elementos do sistema de controle com os elementos de degeneração incorporados. Esses modelos são gerados e organizados segundo a técnica do PFS/MFG (Anexo E) como especializações de diagramas de atividades da UML. Devem retratar os cenários de detecção de falhas e a atuação sobre os elementos da arquitetura do sistema de controle. Nesta modelagem, devem-se incluir as condições de regeneração, ou seja, a recomposição do sistema para operação plena. A Figura 4.12 ilustra um exemplo de modelagem comportamental. Neste exemplo, se ocorrer uma falha no sistema de refrigeração, o tratamento desta falha é acionado e no caso de não conseguir resolver a falha, a degeneração é acionada (desliga-se os segmentos de ar-condicionado menos prioritários e ativa-se os ventiladores).

Figura 4.12. Exemplo: modelo MFG para degeneração no caso de ocorrência de falha no sistema de refrigeração.

2) Componentes

Uma vez modeladas estática e dinamicamente todas as classes, do sistema de controle com os mecanismos de degeneração incorporados deve-se especificar os componentes a serem utilizados. Estes componentes podem ser obtidos através da agregação das classes definidas anteriormente.

- Identificação dos componentes de detecção: devem-se identificar os componentes que indicam as situações críticas e detectam as ocorrências de falhas (ou a iminência de sua ocorrência). Eles também verificam a ausência de falhas, que caracterizam as situações de regeneração. Esses componentes podem ser implementados na forma de agentes;
- Identificação dos componentes de atuação (reconfiguração): após a definição das classes para os elementos de atuação, implementam-se os agentes de atuação que são responsáveis pela mudança na configuração do sistema de modo automático, ou com a participação do operador humano.

A Figura 4.13 apresenta um modelo para criação de um componente para degeneração. Neste caso, agregam-se as classes de supervisão, reconfiguração e Célula de degeneração.

Figura 4.13. Componente para degeneração.

Através dos elementos de controle típicos, pode-se criar o componente de controle, como ilustrado na Figura 4.14. Neste caso, este componente é composto de outros componentes (ou classes): ‘Módulo Sinais Sensoriais’ e ‘Modulo Sinais para Atuadores’.

Figura 4.14. Componente de controle típico.

A modelagem dos componentes é baseada nas técnicas de análise orientada a objetos (OOA²⁰) (BOOCH, 1994) e projeto orientado a objetos (OOD²¹) (BOOCH, 1994; RUMBAUGH, 1994).

4.4.3. Etapa3 - Especificação técnica do Software Controle de Degeneração

Nesta etapa, os componentes de software de supervisão e de reconfiguração são detalhados em termos de suas estruturas e dos algoritmos para que sejam

²⁰ “Object Oriented Analysis”.

²¹ “Object Oriented Design”.

implementados. Neste caso, as especificações são detalhadas com identificação das variáveis e das funções computacionais que caracterizam seus componentes, de modo a subsidiar os desenvolvedores responsáveis pela implementação.

O detalhamento dos componentes aborda os seguintes tópicos (Tabela 4.3):

- **Atributos:** variáveis que implementam a organização dos dados e o domínio de valores válidos;
- **Serviços:** funções disponibilizadas pelos componentes para serem acionados pelo sistema de controle.

Tabela 4.3.Exemplo: especificação técnica da monitoração de pontos críticos.

Subsistema	Atributos	Serviços/operações
Controle de Acesso	<i>_EstadoAcessoControlado:</i> <i>booleano</i> – informa a necessidade da degeneração ou não.	<i>booleano</i> <i>getEstadoAcessoControlado()</i> – verifica o estado do sensor específico para o ponto crítico.

Finalmente, com as especificações de todos os componentes definidos e com a incorporação dos mecanismos de degeneração, as próximas atividades a serem realizadas para completar o processo de desenvolvimento do projeto do software de controle (com degeneração), são a implementação e os testes destes elementos que formam o software do sistema de controle:

- Para a implementação do sistema, consideram-se as especificações dos componentes desenvolvidas nas etapas anteriores, que refletem a incorporação de mecanismos de degeneração.
- Após a implementação e antes de se colocar o sistema produtivo em operação, é necessário que se realizem testes de validações do sistema de controle com os mecanismos de degeneração. Nesta etapa, acontece o ajuste do sistema de controle da degeneração aplicado ao sistema produtivo, considerando as

medições e registros de comportamentos sob situações não desejadas (de falhas ou anormais).

4.5. Comportamento do sistema de controle com os requisitos de degeneração num ambiente de sistema produtivo

Considera-se que após a geração das especificações de um sistema de controle através do método apresentado o sistema é implantado num ambiente e/ou processo produtivo. A Figura 4.15, a seguir, ilustra como é a operação deste sistema no ambiente produtivo no caso de ocorrência de falhas. Quando é identificada a ocorrência de um problema, faz-se a avaliação do impacto deste problema no funcionamento do processo produtivo e, de acordo com esta avaliação, tem-se a reconfiguração da arquitetura do sistema de tal forma que o desempenho do sistema seja mantido ou o impacto neste seja minimizado mantendo-se os objetivos finais de produção ou ainda dependendo do impacto que o problema tem no processo, inicia-se o desligamento gradual de todo o sistema. Este procedimento pode ser automático ou pode envolver a intervenção de um ser humano (operador) para comandar o sistema através informações geradas pelo sistema degenerativo.

Figura 4.15. Sistema degenerativo num sistema produtivo.

4.6. Comentários sobre o capítulo

Este capítulo apresentou as etapas que compõem o método proposto. Inicialmente apresentou-se uma comparação entre o projeto do software do sistema de controle típico e o projeto com a incorporação dos mecanismos de degeneração. Foi identificada a necessidade de considerar mais um ponto de vista (“Requisitos de Controle”) na análise das especificações da arquitetura de software. Foram apresentadas as etapas necessárias. Para obtenção das especificações do software de controle com os requisitos de degeneração. Desta maneira, têm-se os requisitos que permitem a aplicação destas etapas do método cuja aplicação é exemplificada no estudo de caso (Edifícios Inteligentes) abordado no capítulo 5.

5. ESTUDO DE CASO: EDIFÍCIOS INTELIGENTES

Neste capítulo, apresentam-se os Edifícios Inteligentes (EI) como estudo de caso para a aplicação e validação do método proposto. Um EI no contexto deste trabalho é visto como um sistema produtivo que deve assegurar conforto, segurança e outros serviços. Inicialmente, descrevem-se os elementos de um EI e em seguida tem-se um exemplo da aplicação do método para o desenvolvimento do sistema de controle com os requisitos de degeneração neste estudo de caso.

A utilização inicial da expressão “Edifício Inteligente” refere-se a uma proposta de controle e disponibilização dos serviços nos edifícios através dos seus subsistemas como: ventilação e ar-condicionado (HVAC), iluminação, alarmes de incêndio, proteção contra incêndios, controle de acessos, segurança e redução de consumo de energia elétrica. Todos eles gerenciados por um sistema de controle que integra os dispositivos e as operações dos subsistemas. Neste contexto, com a aplicação do método proposto, obtêm-se os softwares de controle para os EI, que além de assegurar a operação em situações previstas, permite aumentar a eficiência e as funcionalidades dos sistemas de controle em situações inesperadas.

Assim sendo, o Edifício Inteligente foi escolhido como estudo de caso para a aplicação deste método, porque ele pode ser caracterizado como Sistema Produtivo, Sistema à Eventos Discretos e Sistema Distribuído contendo vários subsistemas com seus respectivos controles.

5.1. Edifícios Inteligentes e seus subsistemas

Um edifício Inteligente é composto por vários subsistemas (ARKIN e PACIUK, 1995; BOYD, 1994; FLAX, 1998; GOMES, 1998; Anexo C); dentre eles podemos citar (Figura 5.1):

- **Transporte/movimentação** – compostos por elementos que facilitam e organizam o transporte e a movimentação de pessoas e objetos. Podem-se citar os elevadores, as escadas rolantes, as esteiras rolantes, entre outros.
- **Segurança patrimonial e pessoal e controle de acesso** – permite o controle de entrada e saída de pessoas e objetos no interior do prédio. Os principais dispositivos são: catracas eletrônicas, câmeras associadas ao circuito interno de TV, leitores de códigos de barras, leitores de digitais, leitores de íris, portas especiais, alarmes.
- **Gerenciamento de energia** – controla o fornecimento de energia para todos os subsistemas de um edifício.
- **Incêndio** – sistema para identificação, sinalização e controle de ocorrência de incêndio composto por detectores de fumaça, portas corta-fogo e dispositivos automáticos para apagar focos de incêndio.
- **Iluminação** – sistema que garante iluminação adequada e eficiente dos ambientes. Interage com os elementos como: lâmpadas, cortinas, baterias auxiliares.
- **Ventilação e ar condicionado (HVAC)** – sistema responsável para manter os ambientes confortáveis do ponto de vista de ventilação, temperatura e umidade.

Estes subsistemas devem interagir com base em controladores autônomos (agentes) que trocam informações entre si através de redes de comunicação padronizadas.

A Figura 5.1 ilustra a distribuição dos subsistemas no edifício. Cada subsistema deve possuir o seu controle inteligente, autônomo e cooperativo de tal forma a caracterizar um sistema de controle distribuído.

Figura 5.1. Subsistemas em edifícios inteligentes.

Neste caso, cada subsistema possui um componente de controle associado. A Figura 5.2 apresenta um detalhamento do componente de controle. Neste caso, *<subsis>* indica o subsistema correspondente (para os Edifícios Inteligentes, os subsistemas estão relacionados no item 5.1).

Figura 5.2. Componente de controle do subsistema *<subsis>*.

5.2. Edifícios Inteligentes, seus subsistemas e com requisitos de degeneração

A inclusão dos requisitos de degeneração no sistema de controle de EI é ilustrada de forma geral na Figura 5.3. Neste caso, todos os subsistemas possuem uma conexão com o módulo de degeneração global, que é o responsável pela alteração dos requisitos de controle no caso de ocorrência de alguma situação crítica (não resolvida no tratamento de falhas). Essas alterações permitem que o sistema de controle atue com degeneração (mantendo ativas as operações mais importantes) ou com regeneração, retornando ao funcionamento normal.

Cada subsistema apresentado na Figura 5.3 contém um componente de degeneração associado ao componente de controle. A Figura 5.4. ilustra estes elementos que interagem com os módulos de degeneração global e controle global.

Figura 5.3. Subsistemas de edifícios inteligentes com degeneração.

Na Figura 5.4, <subsis> corresponde ao subsistema de edifícios inteligentes que foram relacionados no item 5.1.

Figura 5.4. Componente de controle associado ao componente de degeneração do subsistema <subsis>.

A seguir, aplica-se o método para alguns dos subsistemas do edifício inteligente como: Transporte, Controle de Acesso e Ventilação e Ar-Condicionado (HVAC). No sentido de ilustrar a aplicação do método proposto, utilizamos apenas alguns casos como exemplos visando à implementação de modelos e especificações correspondentes à degeneração.

5.3. Aplicação do Método Proposto – um Exemplo

A partir das descrições do método apresentadas no capítulo 4, aplicam-se aos subsistemas do Edifício Inteligente, as etapas propostas para inclusão dos requisitos de degeneração para este estudo de caso.

5.3.1. Etapa1 - Identificação dos pontos críticos do sistema de controle

Nesta etapa identificam-se os pontos críticos do Edifício Inteligente (considerado como um sistema produtivo). Na Tabela 5.1. apresenta-se como exemplo a relação de alguns Pontos Críticos e as respectivas degenerações que podem ocorrer nos ambientes dos Edifícios Inteligentes.

Tabela 5.1. Alguns Pontos Críticos para o Edifício Inteligente.

	Subsistema	Ponto Crítico	Degeneração
1	HVAC	Fluxo de água fria num sistema de Ventilação e Ar-condicionado. (pode falhar)	<p>É identificado que o tratamento de falha não conseguiu recuperar a operação do ar condicionado;</p> <p>Desliga ou reduz os segmentos menos prioritários;</p> <p>Se não conseguir, aciona sistema de ventiladores de teto nos pontos que perderam o ar-condicionado</p>
2	Controle de acesso	ambiente de acesso seguro. (pode falhar)	Liberar o acesso aos técnicos de manutenção, ou ativar o acesso através de uma entrada especial, pois o serviço não pode parar.
3	Transporte	Sistemas eletrônicos de portas dos elevadores e escadas rolantes. (sobrecarga de energia)	<p>A monitoração verifica as falhas e comunica o sistema de controle para:</p> <ul style="list-style-type: none"> - Desligar alguns elevadores; - Manter portas de escadas destravadas.

5.3.2. Etapa2 - Modelagem dos mecanismos de degeneração

Nesta etapa do método desenvolvem-se os modelos através do diagrama de classes dos elementos que compõem o sistema produtivo.

Modelagem estática:

Deve-se incorporar (acoplar) os procedimentos associados à degeneração ao modelo de sistema de controle. A Figura 5.5 apresenta o diagrama de classe para representar a hierarquia para obtenção dos controles para os subsistemas Transporte, Controle de Acesso e HVAC. Verifica-se que para incluir a degeneração, é necessária a associação, para cada componente de controle, de um componente de degeneração.

Figura 5.5. Diagrama de classes e, em destaque, os componentes para degeneração para cada subsistema.

As figuras 5.6, 5.7 e 5.8 ilustram um detalhamento da associação dos componentes de controle e degeneração para os subsistemas Transporte, Controle de Acesso e HVAC. Observa-se que as estruturas dos diagramas para os três subsistemas são semelhantes, indicando desta maneira uma alta probabilidade de reutilização.

Figura 5.6. Componente de controle associado ao componente de degeneração do subsistema Transporte.

Figura 5.7. Componente de controle associado ao componente de degeneração do subsistema Controle de acesso

Figura 5.8. Componente de controle associado ao componente de degeneração do subsistema HVAC.

Aumentando-se o nível de detalhes para as representações das figuras 5.6, 5.7, 5.8 e considerando somente os requisitos de degeneração obtém-se os diagramas das figuras 5.9, 5.10, 5.11, respectivamente.

Figura 5.9. Modelo de degeneração para o subsistema Transporte.

Figura 5.10. Modelo de degeneração para o subsistema Controle de Acesso.

Figura 5.11. Modelo de degeneração para o subsistema HVAC.

Modelagem dinâmica:

A modelagem dinâmica apresentada neste item corresponde aos pontos críticos 1 e 3 relacionados na Tabela 5.1 do item 5.3.1 (etapa 1) que identifica os pontos críticos do sistema produtivo (edifício inteligente). O desenvolvimento dos modelos dos outros pontos críticos segue procedimento análogo e os respectivos modelos são apresentados posteriormente.

A Figura 5.12 apresenta a dinâmica de funcionamento dos equipamentos para refrigeração e aquecimento dos ambientes. Neste caso, é ilustrada a dinâmica para manter a temperatura ideal num ambiente, para obtenção de água morna, fria e quente e para indicar a quebra de máquinas (falha). Estes modelos são gerados com base nas especificações anteriormente levantadas e com base na metodologia PFS/MFG de construção de grafos (MIYAGI, 1996).

Figura 5.12. Modelo MFG de conforto térmico com o tratamento de falha (quebra de máquina) do subsistema HVAC (VILLANI et al., 1999).

Na Figura 5.12 indicam-se as condições Ligar e Desligar das máquinas (Refrigeração e Aquecedor) que correspondem aos componentes de atuação (atuadores) que ativam os elementos externos (Ligar e Desligar do Aquecedor e da Refrigeração) no MFG da Figura 5.13. As condições ‘Quente’ e ‘Frio’ no ar do ambiente (Figura 5.12) correspondem às informações de sensores de temperatura que possibilitam a ativação do aquecimento ou da refrigeração apresentada na Figura 5.13.

A Figura 5.13 ilustra um MFG que especifica um procedimento de controle da temperatura para que seja mantido num determinado valor desejado. Neste caso, em função de sinais de sensores que detectam as temperaturas (‘Frio’, ‘Quente’) ativa-se ou a atividade de aquecimento ou a atividade de resfriamento (ligando-se os equipamentos correspondentes).

Figura 5.13. Modelo MFG para manter a temperatura desejada para um processo normal.

O modelo MFG da Figura 5.14 demonstra a dinâmica para o acionamento da degeneração no subsistema HVAC, no caso da falha: quebra do sistema de refrigeração.

Neste caso, na degeneração acionam-se os ventiladores até que os ambientes fiquem desocupados, quando a ventilação é desligando. Ou seja, os ambientes continuam em funcionamento mesmo na ocorrência de alguma situação crítica.

Figura 5.14. Modelo MFG para degeneração no caso de ocorrência de falha no sistema de refrigeração.

O modelo MFG da Figura 5.15 demonstra a dinâmica para o acionamento da degeneração no caso da falha no subsistema Controle de Acesso (ponto crítico 2). Neste caso, quando ocorre uma falha num acesso controlado, se o tratamento de falha resulta em “falha não recuperada” na degeneração aciona-se um acesso especial para liberar entrada de técnicos de manutenção.

Figura 5.15. Modelo MFG para degeneração no caso de ocorrência de falha no acesso controlado.

O modelo MFG da Figura 5.16 demonstra a dinâmica para o acionamento da degeneração no caso da falha no subsistema de Transporte devido a uma sobrecarga de energia (ponto crítico 4). Neste caso, na degeneração desligam-se alguns elevadores e destravam-se as portas das escadas.

Figura 5.16. Modelo MFG para degeneração no caso de ocorrência de falha no Transporte devido a uma sobrecarga de energia.

Definição dos componentes:

Os componentes de degeneração (Figura 5.17) e de controle (Figura 5.18) são semelhantes para todos os subsistemas em análise. Ou seja, os modelos dos componentes de degeneração e de controle para outros subsistemas do edifício inteligente são semelhantes aos modelos ilustrados nas figuras 5.17 e 5.18. A única alteração é no nome do subsistema que é definida por *<subsis>*. Neste caso *<subsis>* corresponderá aos subsistemas Transporte, Controle de Acesso e HVAC.

Figura 5.17. Componente para degeneração para <subsis>.

Figura 5.18. Componente para controle de <subsis>.

Outros componentes podem ser obtidos a partir dos elementos tradicionais como ‘Sensores’ (Figura 5.19), ‘Atuadores’.

Figura 5.19. Componente ‘Sensor’ composto por ‘Sensor digital’ e ‘Sensor analógico’.

5.3.3. Etapa3 - Especificação técnica do Software Controle de Degeneração

Os atributos e os serviços/operações das classes de degeneração são detalhados nesta etapa. Ou seja, apresentam-se as especificações técnicas dos componentes de supervisão de pontos críticos e de reconfiguração para os subsistemas Controle de Acesso, HVAC e Transporte.

5.3.3.1 Especificação técnica de supervisão de pontos críticos

Este componente está associado com o componente de controle. Assim, ele acessa as mesmas informações oferecidas pelos componentes ‘sensores’ localizados nos componente de controle. Na Tabela 5.2 são apresentadas as especificações para este componente.

Tabela 5.2.Especificação técnica da supervisão de pontos críticos.

Subsistema	Atributos	Serviços/operações
Controle de Acesso	<i>_EstadoAcessoControlado:</i> <i>booleano</i> – informa a necessidade da degeneração ou não.	<i>booleano getEstadoAcessoControlado()</i> – verifica o estado do sensor específico para o ponto crítico.
HVAC	<i>_EstadoBombaCritica:</i> <i>booleano</i> – informa a necessidade da degeneração ou não. <i>_EstadoArCritico:</i> <i>booleano</i> – informa a necessidade da degeneração ou não.	<i>booleano getEstadoBombaCritica()</i> – verifica o estado do sensor específico para o ponto crítico. <i>booleano getEstadoArCritico()</i> – verifica o estado do sensor específico para o ponto crítico.
Transporte	<i>_EstadoEnergiaTranspCritico:</i> <i>booleano</i> – informa a necessidade da degeneração ou não	<i>booleano</i> <i>getEstadoEnergiaTranspCritico ()</i> – verifica o estado do sensor específico para o ponto crítico.

5.3.3.2 Especificação técnica de reconfiguração

Este componente está associado com o componente de controle. Assim, ele envia informações para os componentes ‘atuadores’ localizados nos componente de controle. Desta maneira, a degeneração correspondente é realizada ativando os componentes de atuação correspondentes (Tabela 5.3).

Tabela 5.3. Especificação técnica da reconfiguração.

Subsistema	Atributos	Serviços/operações
Controle de Acesso	<i>_idAcessoControlado: inteiro</i> – informa o identificador do acesso controlado.	<i>ativaAcessoControlado(idAcessoControlado: inteiro)</i> – envia informação para o controle ativar o acesso especial.
HVAC	<i>_idSegmentosAr[]: inteiro</i> – informa o identificador dos segmentos de ar-condicionado. <i>_idVentiladores[]: inteiro</i> – informa o identificador de ventiladores	<i>desativaSegmentosAr(idSegmentosAr[]: inteiro)</i> – envia informação para o controle desativar os segmentos menos prioritários. <i>ativaVentiladores((idVentiladores[]: inteiro)</i> – envia informação para o controle ativar os ventiladores.
Transporte	<i>_idElevadores[]: inteiro</i> – informa o identificador dos elevadores. <i>_idEscadasRolantes[]: inteiro</i> – informa o identificador das escadas rolantes. <i>_idPortasEscadas[]: inteiro</i> – informa o identificador das portas das escadas.	<i>desativaElevadores(idElevadores[]: inteiro)</i> – envia informação para o controle desativar os elevadores. <i>ativaPortasEscada((idPortasEscadas[]: inteiro)</i> – envia informação para o controle ativar as portas das escadas.

5.4. Comentários sobre o capítulo

Este capítulo ilustra a aplicação das etapas de projeto que incluem os requisitos de degeneração no estudo de caso Edifício Inteligente. Os exemplos de subsistemas analisados neste estudo de caso foram: de Transporte, de Controle de Acesso e de ventilação e ar-condicionado (HVAC). Assim, foram gerados alguns exemplos dos artefatos de software representados pelos modelos de classes, modelos de componentes e as especificações relacionadas com a degeneração.

6. CONCLUSÕES

Foram apresentados os principais conceitos envolvidos no desenvolvimento de forma sistemática de softwares para controle de sistemas produtivos com os requisitos de degeneração. O método proposto explora assim os pontos positivos de várias abordagens para o projeto de sistemas de controle com as características de degeneração (redução gradual dos níveis de serviços). Neste sentido, aumenta-se o grau de proteção do sistema contra eventuais situações indesejadas como a interrupção abrupta no caso da ocorrência de situações de falha.

A incorporação deste método gera especificações de software de controle que, por construção, atende às necessidades desejadas para o projeto.

Considerando ainda que existe uma necessidade crescente dos subsistemas que compõem os sistemas produtivos serem autônomos e inteligentes, foram adotadas as técnicas de orientação a objetos e componentização que possibilita um direcionamento, através das especificações resultantes, para a implementação de agentes para os Sistemas Multi Agentes e/ou os hólons para os Sistemas Holônicos. Além disso, os resultados do trabalho permitem que os conceitos de arquiteturas multi-camadas (“n-tier”) também podem ser incorporados. Neste caso, o desenvolvimento de um “middleware” visando a reconfiguração do sistema tem assim papel relevante no projeto do software de controle com os requisitos de degeneração, pois muitas vezes estes sistemas produtivos possuem

um alto grau de heterogeneidade de subsistemas que necessitam trocar informações entre si (interoperabilidade).

Além disso, a aplicação do método proposto em outros sistemas (de Manufatura, Financeiros, Aeronáuticos, entre outros) permitirá a avaliação e validação destes procedimentos nestes sistemas. Neste caso, se estes sistemas forem caracterizados como Sistemas Produtivos (como definido no texto), a sistemática apresentada pode ser aplicada de maneira consistente.

6.1. Trabalhos Futuros

A partir deste método, é possível o desenvolvimento de aplicações que gerem automaticamente as especificações de um sistema de controle baseado em degeneração. Neste sentido, propõem-se os seguintes trabalhos para investigação relacionados com este tema:

- Aplicação prática em outros casos no sentido de avaliação dos resultados e validação dos procedimentos.
- Desenvolvimento de aplicativo (por ex: um simulador de operações degenerativas) para auxiliar na análise e implementação do software de controle de sistemas produtivos.
- Implementação de um protótipo baseado em controladores programáveis (CP) com as características de degeneração incorporadas.
- Desenvolvimento de componentes genéricos de software contendo as características de degeneração para serem aplicados em diferentes tipos sistemas produtivos.
- Desenvolvimento de um “framework” padrão para software de controle de sistemas produtivos com as características de degeneração incorporadas.
- Especificação e implementação de Sistemas Multi Agentes contendo as características de degeneração.

ANEXO A – NORMA ISO/IEC 9126

As organizações ISO/IEC disponibilizam uma norma que relaciona os requisitos necessários para o desenvolvimento de artefatos de software com requisitos de qualidade. Esta norma é conhecida através do código ISO/IEC 9126.

A.1. Qualidade de produtos de software

A Tabela A.1 apresenta as características e sub-características da qualidade de produtos de software (ISO/IEC 9126, 1998).

Tabela A.1. Características e sub-características para a qualidade de software.

Características	Sub-características
Funcionalidade: as funções e propriedades específicas do produto satisfazem as necessidades do usuário.	Adequação: existência de um conjunto de funções apropriadas para as tarefas requeridas. Acurácia: produção de resultados ou efeitos corretos. Interoperabilidade: habilidade de interação do produto de software com outros produtos. Conformidade: o produto está de acordo com as convenções, as normas e os regulamentos estabelecidos. Segurança: aptidão para evitar acessos não autorizados a programas e dados.
Confiabilidade: o produto de software é capaz de manter seu nível de desempenho, ao longo do tempo, nas condições estabelecidas.	Maturidade: estado de maturação do software, detectada por sua baixa frequência de falhas Tolerância a falhas: o nível de desempenho é mantido quando ocorrem falhas Recuperabilidade: existem mecanismos que restabelecem e restauram os dados após a ocorrência de falhas.

Usabilidade: esforço necessário para a utilização do sistema, baseado em conjunto de implicações e de condições do usuário.	Inteligibilidade: facilidade de entendimento dos conceitos utilizados no produto de software Apreensibilidade: facilidade de aprendizado de software mantido quando ocorrem falhas Operacionalidade: faculdade de operar e controlar operações pertinentes ao software.
Eficiência: os recursos e os tempos envolvidos são compatíveis com o nível de desempenho requerido pelo software.	Comportamento no tempo: refere-se ao tempo de resposta de processamento Comportamento dos recursos: relaciona-se com a quantidade dos recursos empregados.
Manutenibilidade: refere-se ao esforço necessário para a realização de alterações específicas, no produto de software.	Analísabilidade: característica de ser possível diagnosticar deficiências e causas e falhas Modificabilidade: característica que o produto deve ter de forma a facilitar modificações e remoções de defeitos Estabilidade: ausência de riscos ou ocorrências de defeitos inesperados no software Testabilidade: facilidade do produto ser testado.
Portabilidade: facilidade do software ser transferido de um ambiente para outro.	Adaptabilidade: faculdade do produto ser adaptado a novos ambientes Instalabilidade: facilidade de instalação do produto de software Conformidade com padrões de portabilidade: o produto esta de acordo com os padrões ou convenções de portabilidade. Substituibilidade: o produto de software pode ser substituído por outro, sem grandes esforços

ANEXO B - MODELAGEM DE SISTEMAS ORIENTADOS

A OBJETOS

A modelagem é fundamental em praticamente todas as áreas da engenharia. A modelagem considera quatro princípios básicos:

- Escolha do modelo que será criado tem uma profunda influência em como o problema será atacado e como a solução será montada.
- Os modelos podem expressar dados em diferentes níveis de precisão.
- Os melhores modelos são aqueles conectados com a realidade.
- Nenhum modelo simples é o suficiente. Todo sistema não-trivial é mais bem enfocado através de um conjunto de modelos relacionados.

B.1. Meta modelo UML

A UML define uma **notação** e um **meta-modelo** (FOWLER e SCOTT, 2003) para a descrição dos elementos e seus inter-relacionamentos.

O **meta-modelo** é um diagrama (usualmente diagrama de classes) que representa uma **notação**. Permite melhorar o rigor da representação, sem sacrificar a sua usabilidade²².

²² Facilidade de utilização. Neste caso, de se modelar.

Figura B.1. Meta-modelo UML, representando a associação e a generalização (adaptado de FOWLER e SCOTT, 2003).

A **notação** é o conjunto de elementos gráficos que são utilizados nos modelos. É a sintaxe da linguagem de modelagem. Por exemplo, a notação ‘diagrama de classes’ define como os itens e os conceitos (classes, associações e a multiplicidades) são representados.

Na Figura B.1 é ilustrado um exemplo de meta-modelo, utilizando um diagrama de classes que representa a **notação** para os conceitos de herança e associação.

A UML unifica e formaliza os métodos de muitos enfoques OO. A Figura B.2 mostra como surgiu a UML. Basicamente, ela foi criada para auxiliar na análise e projeto de sistemas de software. Os blocos representam as várias técnicas existentes no período de 1990 até 1995, onde se podem destacar os trabalhos de Booch, Rumbaugh e Jacobsen que mais tarde se reuniram e criaram uma representação única (1ª versão da UML). Em 1999 houve a criação do UP (“Unified Process”) que padronizava as notações existentes. Finalmente, a partir de 2000 foram criadas as versões de UML padronizadas pela ISO.

Figura B.2. Histórico de surgimento da UML (adaptado de OESTEREICH, 1999).
Evolução das técnicas propostas e respectivos autores.

B.2. Modelo Conceitual da UML

A definição do modelo conceitual da UML é feita através de características como os “building blocks” que são relacionados a seguir:

- **Elementos**

- estruturais: são nomes de modelos UML. É a parte estática de um modelo, representando elementos conceituais ou físicos. Existem sete elementos estruturais que são: (1) *class* é a descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semânticas. (2) *interface*, é uma coleção de operações que especifica um serviço de uma *class* ou *component*. (3) *collaboration*

que define uma interação e é uma associação de papéis e outros elementos que trabalham juntos para prover alguns comportamentos cooperativos. (4) *use case* é uma descrição do conjunto de seqüência de ações que um sistema realiza e gera um resultado observável de valores particulares para um ator (“actor”). (5) *active class* é um objeto que pode iniciar uma atividade de controle. (6) *component* é uma parte física e substituível que possibilita a realização de um conjunto de interfaces. (7) *node* é um elemento físico que existe em tempo real e representa um recurso computacional. Normalmente, tem memória e certa capacidade de processamento.

- comportamentais: são as partes dinâmicas do modelo UML. O elemento *interaction* é um comportamento de um conjunto de mensagens trocadas entre um conjunto de objetos num contexto particular para alcançar um propósito específico. O outro elemento é o *state machine* que é um comportamento que especifica a seqüência de estados que um objeto ou interação irá percorrer durante seu ciclo de vida em resposta à ocorrência de eventos.
- de agrupamento: são partes organizacionais dos modelos UML. O *package* é um mecanismo de propósito geral para organizar elementos em grupos.
- de anotações: são partes do modelo UML para explanação, isto é, para contribuir no entendimento dos modelos. Um *note* é simplesmente um símbolo para incluir comentários associados a um elemento ou a uma coleção de elementos.

- **Relacionamentos**

- Dependência: é uma relação semântica entre duas coisas no qual, uma mudança em uma delas pode afetar a semântica da outra. Graficamente é representado pela Figura B.3.

Figura B.3. Relação de dependência.

- Associação: é uma relação estrutural que descreve uma ligação entre objetos. A agregação é um tipo de associação que representa a relação entre um todo e suas partes. Graficamente, é representado pela Figura B.4.

Figura B.4. Relação de associação. O sistema HVAC possui um ou mais sensores.

- Generalização: é uma relação de especialização ou generalização onde os objetos de elemento especializado (filho) herdam atributos de elementos generalizados (pai). Graficamente, é representado pela Figura B.5.

Figura B.5. Relação de Generalização.

- Realização: é a relação semântica entre as interfaces e as classes ou componentes que implementam as interfaces. Graficamente é representado pela Figura B.6.

Figura B.6. Relação de Realização.

B.3. Diagramas UML

Um modelo em UML é constituído por um conjunto de diagramas (representação gráfica) consistentes entre si e descrições textuais de elementos (por ex. classes) que podem aparecer em diferentes diagramas. No UML existem 9 diagramas padronizados (Figura B.7) que são divididos em:

- Diagramas para representação estática: casos de uso (“use case”), classes, objetos, componentes e distribuição (“deployment”).
- Diagramas para representação dinâmica: sequência, colaboração, estados (“statechart”) e atividades

Figura B.7. Os diagramas da UML.

B.3.1. Diagrama de Casos de Uso (“use case”)

Permite a representação do sistema de acordo com a visão dos usuários. Normalmente é construído no início do desenvolvimento do projeto por especialistas do domínio do problema. Tem como objetivos: especificar o contexto do sistema, capturar os requisitos do sistema, validar a arquitetura e direcionar a implementação e os testes (Figura B.8).

Figura B.8. Diagrama de Caso de Uso (adaptado de BOOCH, 1999).

B.3.2. Diagrama de Classes

Permite a modelagem dos elementos e conceitos de um sistema. Especifica os esquemas lógicos dos dados e seus inter-relacionamentos. Normalmente é construído e refinado ao longo do desenvolvimento do projeto (Figura B.9).

Figura B.9. Diagrama de Classes (adaptado de BOOCH, 1999).

B.3.3. Diagrama de Objetos

Esses diagramas ilustram a estrutura de objetos (instâncias de classes) e as ligações (instâncias de associações). Normalmente construído na etapa de análise e projeto do sistema (Figura B.10).

Figura B.10. Diagrama de Objetos (adaptado de BOOCH, 1999).

B.3.4. Diagrama de Componentes

Ilustra a estrutura física da implementação. É uma parte da especificação da arquitetura do sistema. Organiza os elementos como um componente que poderá ser utilizado em outros projetos. (Figura B.11).

Figura B.11. Diagrama de Componentes (adaptado de BOOCH, 1999).

B.3.5. Diagrama de Distribuição (“deployment”)

Esse diagrama ilustra a topologia de hardware do sistema. É utilizado como parte da especificação da arquitetura do sistema. Especifica a distribuição dos componentes e o “gargalo” de desempenho (Figura B.12).

Figura A.12. Diagrama de Distribuição (adaptado de BOOCH, 1999).

B.3.6. Diagrama de Seqüência

Permite a representação do comportamento dinâmico do sistema orientado pelo tempo. Ilustra a modelagem do fluxo de controle através de cenários típicos. (Figura A.13).

Figura B.13. Diagrama de Seqüência (adaptado de BOOCH, 1999).

B.3.7. Diagrama de Colaboração

Permite a representação do comportamento dinâmico do sistema orientado pelas mensagens. Ilustra a coordenação entre a estrutura de objetos e o controle (Figura B.14).

Figura B.14. Diagrama de Colaboração (adaptado de BOOCH, 1999).

B.3.8. Diagrama de Estados (“Statechart”)

Permite a representação do comportamento dinâmico do sistema orientado a eventos. Ilustra a modelagem do ciclo de vida dos objetos (interface com o usuário, dispositivos) (Figura B.15).

Figura B.15. Diagrama de Estados (adaptado de BOOCH, 1999).

B.3.9. Diagrama de Atividades

Permite a representação do comportamento dinâmico do sistema orientado a atividades. Ilustra a modelagem do processo de negócios e o “workflow”. Modela os processos (algoritmos) (Figura B.16).

Figura B.16. Diagrama de Atividades (adaptado de BOOCH, 1999).

B.4. MAML (“Multi Agent Modeling Language”)

A UML é bastante utilizada para a modelagem de sistemas de software baseados em objetos. Assim, com base nas características positivas deste padrão, SILVA e LUCENA (2003) introduziram uma extensão da UML para ser utilizada na modelagem de SMA. Neste caso, devido a não existência de um conjunto de abstrações comuns para a descrição dos aspectos estáticos e dinâmicos dos SMA, foi desenvolvido um meta-modelo (“conceptual framework”) que expressa os aspectos mais relevantes destes sistemas. O meta-modelo é conhecido como TAO (“Taming Agents and Objects”) que possui uma ontologia que associa as abstrações já conhecidas como os objetos e classes para outras abstrações como agentes, papéis (funções) e organizações, que juntos, estabelecem a relação entre agentes e objetos da Engenharia de Software. O TAO meta modelo define os aspectos estáticos e dinâmicos dos SMA. Os aspectos estáticos estão relacionados com a definição das entidades (características estruturais e comportamentais) e os relacionamentos entre elas. As entidades definidas em TAO são

objetos, agentes, organizações, funções (objeto-função e agente-função), ambientes e eventos. O aspecto dinâmico do TAO descreve a interação entre seus elementos estáticos. Eles podem ser classificados como processos dinâmicos primitivos (elementares) e processos dinâmicos de alto nível. Os processos dinâmicos primitivos descrevem as interações básicas entre dois elementos. Por exemplo, o processo de criação e de destruição de elementos dos SMA é caracterizado como processos primitivos. Processos dinâmicos de alto nível são comportamentos mais complexos que são descritos através de processos dinâmicos primitivos e outros processos dinâmicos de alto nível. Por causa do conjunto de diferentes elementos e de seus relacionamentos definidos através do TAO, novos diagramas (diagrama de organização e diagrama de função) foram criados e os diagramas existentes da UML (diagrama de classes e diagrama de seqüência) sofreram algumas adaptações.

B.5. Perfil UML

Em FONTOURA et al. (2002) é proposta um perfil UML (denominada UML-F) para auxiliar a descrição de arquiteturas baseadas em “frameworks”. Neste caso, a Figura B.17 mostra a hierarquia entre a UML padrão e as possíveis extensões como UML-F e UML-RT (para sistemas em Tempo Real), entre outros.

Figura B.17. Representação hierárquica e as possíveis extensões do padrão UML, através do diagrama de “package”.

ANEXO C - EDIFÍCIOS INTELIGENTES

Segundo o dicionário de língua portuguesa, o adjetivo “inteligente” é dado para tudo aquilo que possui a habilidade ou capacidade de compreender e adaptar-se facilmente. Ou seja, é uma qualidade de se entender as condições atuais e uma capacidade de prover respostas apropriadas de acordo com estas condições. Desta maneira, pode-se descrever um edifício inteligente como sendo aquele que tem a capacidade de se adaptar de acordo com as mudanças ocorridas no seu ambiente, sempre direcionado para uma melhoria na sua usabilidade (conforto aos usuários e facilidades de manutenção pelos operadores e gerentes), com custos reduzidos.

Este conceito de inteligência atribuída aos edifícios não é novo pois, desde os primórdios dos primeiros abrigos utilizados pelo ser humano, estes eram sempre “projetados” de tal forma que os elementos que formavam estes ambientes procuravam ser integrados com o intuito de melhorar cada vez mais o conforto de seus ocupantes, tanto para moradia como também para local de trabalho.

Existem diferentes tipos de edifícios como comerciais, educacionais, médicos e residenciais. Todos eles com suas funções específicas que podem ser melhoradas de tal forma que possam ser considerados “inteligentes”. Porém, as funções de “inteligência” que devem ser atribuídas a eles variam de acordo com a sua funcionalidade. Ou seja, algumas funções possuirão características de inteligência intensificadas de acordo com a

sua funcionalidade básica. Como exemplo de tipos de edifícios e suas funcionalidades, podemos citar:

- Edifício hospitalar com funções relacionadas ao atendimento médico;
- Edifício comercial com funções para melhoria do atendimento aos seus clientes.
- Edifício escolar com funções para melhoria das atividades de ensino.

Em meados do século 20, o termo “inteligente” foi atribuído aos prédios, não devido a alguma habilidade para produzir necessidades culturais ou ambientais, mas sim como um apelo de marketing para promover um prédio que estava dotado das últimas tecnologias de computação e de automação.

Nas publicações relacionados com este tema verifica-se a existência de diversos termos que em princípio possuem mesmos significados, como: automação predial, edifícios inteligentes, “building automation”, “smart buildings” e domótica; este último relacionado inicialmente com a construção de moradias com diversos sistemas integrados e automatizados. Todos estes termos estão associados com a construção dos ambientes de forma integrada e com características de inteligência. Dentro deste contexto, GOMES e STEIGER-GARÇÃO (1996), introduzem os *domots*²³ que são entidades que podem gerenciar (controlar) algum tipo de recurso de um prédio. A domótica tem ampliado seu escopo e atualmente considera-se que está relacionada com o estudo para a busca de metodologias que visam a integração dos subsistemas existentes no prédio como: comunicação, segurança, controle e gestão de energia através da interação entre os setores tecnológicos, sociais e econômicos, todos em constante evolução (ANGEL e FRAIGI, 1993).

O desenvolvimento das pesquisas sobre a construção dos chamados edifícios inteligentes envolve uma grande interação entre profissionais de diferentes áreas, todos atuando sobre temas multidisciplinares, envolvendo engenharia, psicologia, economia, arquitetura, entre outros. Este problema não é dirigido ou dominado por um grupo em competição com outros grupos mas, ao contrário, ele é importante para fomentar as pessoas envolvidas para a atuação de forma conjunta na escolha e uso de diferentes

²³ Domots = “domus”+ “robot”

tecnologias e também como uma forma de fertilização de idéias e aplicações em diversas áreas.

C.1. O que é um edifício Inteligente?

De maneira geral, podemos definir um edifício inteligente da seguinte forma: “Um prédio que responde a todos os requerimentos de seus ocupantes”. A implicação desta afirmação é bastante variada. E, certamente milhares de pontos de vistas e explicações podem ser considerados, todas elas corretas de acordo com o contexto.

Desta forma, FLAX (1991) define que um edifício inteligente é aquele que cria um ambiente que maximiza a eficiência dos seus ocupantes e ao mesmo tempo tem um gerenciamento eficiente de recursos efetivos com custos mínimos.

Para se considerar outras afirmações sobre os edifícios inteligentes, outras propostas são apresentadas e examinadas. Cada uma destas definições possui tanto visões filosóficas quanto técnicas com ênfases no desenvolvimento em engenharia e arquitetura. A maioria destas afirmações surgiu ao longo do século 20 juntamente com a grande evolução tecnológica deste mesmo período.

Quando se fala em edifícios inteligentes deve-se levar em conta todos os elementos arquitetônicos a eles associados e a tecnologia. Com isso, sobre a arquitetura, pode-se afirmar que uma arquitetura inteligente é descrita como sendo aquela que é sensível, ou seja, os componentes da arquitetura do prédio são adaptáveis de tal forma que eles possam ser trocados e/ou modificados assim que a utilização do prédio se modifique. Em relação às tecnologias, pode se afirmar que as soluções inteligentes são aquelas tecnologias que são sensíveis a estas mudanças. Assim, a arquitetura e as tecnologias inteligentes, quando juntas, fornecem todas as características básicas para que o prédio seja descrito em termos de suas “capacidades” e “funcionalidades”, das habilidades inerentes ao sistema arquitetônico, das estruturas e das tecnologias associadas. De tal maneira que os edifícios inteligentes possam atender às aspirações dos proprietários e da equipe de projeto de acordo com as especificações do projeto do edifício.

Existem diferentes definições para edifícios inteligentes. De uma forma genérica, estas definições são diferenciadas. Por exemplo, nos EUA o conceito de edifício inteligente está relacionado com o modo como são realizadas as interconexões dos subsistemas de serviço (“service systems”) em benefício dos seus ocupantes. Na Europa, enfatiza-se a interação entre os sistemas e os seus elementos estruturais (estrutura física). No Japão, o enfoque foi direcionado para a utilização de novas e avançadas tecnologias para transferência de dados/informações de forma a melhorar os aspectos organizacional-supervisional.

Segundo ARKIN e PACIUK (1995), algumas das principais definições que procuram descrever o significado dos edifícios inteligentes são:

- Definição dada pelo Instituto de Edifícios Inteligentes (“IBI - Intelligent Building Institute”) dos EUA: “Um prédio inteligente é aquele que possui um ambiente produtivo e de custo efetivo através da otimização dos quatro elementos básicos: estrutura, sistemas, serviços, manutenção e a inter-relação entre eles. A principal característica que todos os prédios inteligentes devem ter em comum é uma estrutura projetada para acomodar mudanças com custos reduzidos”
- Definição do grupo de Edifícios Inteligentes da Europa (“EIBG - European Intelligent Building Institute Group”): “Um prédio inteligente cria um ambiente que aloca a organização de forma que seus escritórios e outras instalações sejam objetivos e maximizam a efetividade de seus ocupantes e ao mesmo tempo aloca uma eficiente manutenção de recursos com um mínimo custo de vida/tempo”.

Já STUBBINGS (1988) limitou a sua definição de edifício inteligente como sendo um prédio que controla totalmente o seu próprio ambiente. A implicação desta definição é a necessidade de existência de um domínio técnico completo de todos os elementos componentes do edifício.

Outra definição dada por MCCLELLAND (1988), para os Edifícios Inteligentes, é mais genérica. Ele afirma que um Edifício Inteligente pode ser o nome dado para aqueles edifícios que tenham sido projetados utilizando-se de maneira coordenada as tecnologias

de controle de processos e de comunicação de dados, com o objetivo de melhorar a distribuição dos recursos disponíveis para a execução de todas as atividades do edifício.

A definição considerada por JUGEND (1996) como uma das mais abrangentes é aquela que descreve um edifício inteligente como sendo um empreendimento imobiliário de qualquer porte, dotado de controle do conjunto de suas funções, ou de parte delas, por sistemas computadorizados, com equipamentos distribuídos ao longo do prédio, interligados a uma central de controle. Estes sistemas são denominados sistemas digitais distribuídos, ou sistemas de supervisão e controle, e têm a função de maximizar as condições operacionais e de manutenção do prédio, minimizando e otimizando os custos envolvidos.

Verifica-se, portanto que existem inúmeras definições para Edifício Inteligente, que se diferenciam de acordo com o ponto de vista e da área de atuação de quem faz estas definições. Percebe-se que todas elas buscam sempre o mesmo objetivo básico que é o de melhorar o conforto dos usuários e a redução dos custos de gerenciamento e manutenção do edifício.

A realização de um edifício inteligente requer uma participação de profissionais de diferentes campos de atuação: Especialista em automação, construção civil, instalação elétrica, instalação hidráulica, telecomunicações e outros profissionais de áreas como arquitetura, estruturas e geotécnica. Além disso, este tipo de empreendimento envolve também profissional que analisam a integração de fatores como: sócio-econômico e questões ecológicas, considerações terapêutica/ambiental que afetam de forma considerável a característica de “inteligência” do edifício.

Uma solução técnica e usual é projetar de acordo com as necessidades do usuário e com a sua utilização de forma prática. Entretanto, uma grande quantidade de equipamentos não garante que o prédio é inteligente. Um prédio inteligente inclui características como: modificável, estruturalmente ativo, capaz de uma integração estrutural e funcional, informativo, interativo, seguro, confortável e orientado à serviços, “healthy” e terapêutico, econômico e produtivo.

Segundo BOYD (1994), o Edifício Inteligente é um conceito e uma área de pesquisa e desenvolvimento que surgiu juntamente com o crescimento significativo da tecnologia de microelectrônica e dos computadores. Os computadores e a tecnologia de informação foram os grandes responsáveis pelas mudanças ocorridas a partir dos anos setenta. Vale lembrar que, até esta época, o comércio e a indústria (inclusive os edifícios) eram administrados sem a utilização destas tecnologias e, atualmente, verifica-se que praticamente todas as atividades relacionadas ao lazer, trabalho e aos edifícios são na sua maioria gerenciadas e/ou mantidas através de computadores e outros dispositivos mecatrônicos integrados por meio das tecnologias de informações.

O conceito de Edifícios de Inteligentes renovou bastante o interesse em edifícios devido exatamente à associação com as tecnologias mecatrônica e de computadores. Embora, isto esteja geralmente associada com os edifícios comerciais, existem aplicações para todos os tipos de edifícios. Esse conceito também está sendo explorado utilizando um outro enfoque, o de Inteligência Artificial. Neste caso, a idéia é a utilização dos computadores para emular o processo de aprendizado de seres humanos através do desenvolvimento de técnicas de aprendizagem e de tomadas de decisões. Na Figura C.1 são apresentados os três possíveis enfoques de definição de edifícios inteligentes.

Figura C.1. Definição de edifício Inteligente.

O enfoque 1 considera um edifício inteligente devido a automação e a utilização de novas tecnologias, o enfoque 2 devido a utilização de Inteligência Artificial, Sistemas

Fuzzy e Redes Neurais e o enfoque 3 correspondem a uma definição utilizando os enfoques 1 e 2 (integrado).

C.2. Edifícios Inteligentes x Sistemas Híbridos

Como o próprio nome diz, híbrido corresponde à heterogeneidade, mistura ou composição de entidades ou processos com características diferentes. Partindo desta afirmação, podemos afirmar que, na natureza, praticamente todos os sistemas dinâmicos podem ser considerados sistemas híbridos.

As aplicações mais modernas de sistemas produtivos dinâmicos são naturalmente bastantes complexas devido à utilização de diferentes subsistemas com diferentes tipos de comportamentos. Dentre as infinitudes de sistemas com estas características podemos citar: sistemas de controle complexos de tempo-real para processamento de sinais, para interfaces homem-máquina em automação, para controle de visão em robótica e, evidentemente, para os edifícios inteligentes. Estas aplicações possuem características particulares, a dinâmica e a presença de eventos contínuos e os eventos discretos simultaneamente. Ou seja, podemos considerar estes sistemas como sendo sistemas híbridos dinâmicos e reativos e cujo comportamento de interesse é determinado pela interação da dinâmica dos componentes discretos e contínuos. ANTISAKLIS (1998) afirma que estes sistemas possuem variáveis e sinais que assumem valores de um conjunto contínuo (o conjunto de números reais) e também variável que assumem valores, normalmente finitos, de um conjunto discreto (o conjunto de símbolos {a, b, c}). Estas variáveis ou sinais contínuos ou discretos dependem de variáveis independentes, como o tempo, que também podem ser contínuos ou discretos; algumas dessas variáveis podem também ser dirigidas a eventos discretos e assíncronas. Um exemplo de um sistema de controle híbrido é um sistema de chaveamento onde o comportamento dinâmico de interesse pode ser adequadamente descrito por um número finito (pequeno) de modelos dinâmicos que são tipicamente conjunto de equações diferenciais e um conjunto de regras para chaveamento entre estes modelos. Estas regras de chaveamento podem ser descritas por expressões lógicas ou uma ferramenta de

representação de sistema a eventos discretos como um autômato finito ou uma representação por rede de Petri.

Atualmente, existem diversas pesquisas sendo realizadas com o enfoque direcionado para os sistemas híbridos. O estudo de propriedades (estabilidade) dos sistemas dinâmicos descritos através de equações diferenciais com presença de descontinuidade é uma das conseqüências provenientes dos sistemas híbridos.

Segundo CHAMPAGNAT (1996), uma das principais motivações deste grande interesse está na necessidade de desenvolvimento de controle supervisorio para processos produtivos contínuos e também a necessidade de substituição de processos mono-produto para processos flexíveis de multi-produto com grandes demandas. O controle supervisorio e a monitoração destes processos produtivos contínuos são geralmente baseados na detecção da ocorrência de eventos que acarretam as mudanças de estados discretos num tempo discreto.

O estudo de sistemas de controle híbrido é muito importante para o projeto de controladores supervisorios dos sistemas contínuos e também para o projeto de sistemas de controle inteligentes com alto grau de autonomia (VALETTE, 1995); (ANTISAKLIS, 1998); (KOWALEWSKI, 1997). Neste caso, este enfoque é utilizado para realizar uma organização hierárquica dos sistemas de controle complexos. Nestes sistemas, esta hierarquia auxilia na administração da complexidade, onde os níveis mais altos na hierarquia requerem modelos menos detalhados (abstrações discretas) do que os níveis mais baixos.

C.3. Conforto Ambiental nos Edifícios Inteligentes

Num Edifício Inteligente, o fator imprescindível para garantir o status de Inteligente é a garantia de conforto ambiental aos ocupantes e usuários destes ambientes. Neste sentido, deve-se assegurar que o subsistema de ventilação e ar-condicionado (HVAC) funciona sempre de forma segura e com eficiência.

O sistema de ventilação e ar-condicionado é utilizado para obtenção do conforto térmico nos ambientes de edifícios (Figura C.2). Nestes sistemas, o fluxo de ar, o aquecimento e

a refrigeração do ar devem ser controlados de tal maneira que seja obtido uma temperatura confortável aos usuários destes edifícios. Normalmente, no interior destes ambientes existem diferentes fontes que geram calor, como os seres humanos (fluxos de pessoas que entram e saem de um ambiente + pessoas trabalhando no ambiente) e os equipamentos (computadores, fax, ventiladores, telefone, equipamentos diversos, etc.) além de paredes e outros elementos por onde o calor externo é transmitido ao ambiente. Um sistema de controle para conforto térmico destes ambientes utiliza sensores de temperatura, termostatos e os sistemas para ventilação e ar-condicionado. Neste caso, o termostato pode ser considerado como um sistema assíncrono, dirigido pela ocorrência de eventos instantâneos e que possui estados discretos como: ‘ambiente quente’, ‘ambiente frio’ e ‘ambiente normal’. A temperatura de um determinado ambiente é interpretada em um destes estados através de um termostato, isto é, a partir destes estados, o termostato tem como saída sinais de corrente elétrica que serão processados no controle do sistema de ventilação e ar-condicionado de modo que o ambiente mantenha a temperatura desejada. Na Figura C.2 verifica-se que a água resfriada ou aquecida é utilizada para condicionar o ar de uma determinada área onde se deseja controlar a temperatura do ambiente.

Figura C.2. Subsistema de Ventilação e Ar condicionado (HVAC).

Desta forma tem-se um sistema produtivo cujo comportamento pode ser estudado como um sistema híbrido, isto é, envolvendo variáveis discretas relacionadas com o estado desejado de conforto e as variáveis contínuas relacionadas com temperatura, velocidade do ar, fluxo de água, etc.

No caso específico do presente trabalho, considera-se a interface com o sistema de HVAC como um sistema a eventos discretos, isto é, a parte híbrida é vista como uma característica interna do HVAC.

ANEXO D - REDES DE PETRI

Inicialmente, as Redes de Petri foram utilizadas para a modelagem e a visualização das características como concorrência, sincronização e compartilhamento de recursos. Estas redes são bastante utilizadas, principalmente em sistemas dinâmicos a eventos discretos (SED). Uma boa introdução sobre as redes de Petri pode ser encontrada em MURATA (1989) e REISIG (1992). A rede de Petri é uma ferramenta que possibilita uma análise qualitativa e quantitativa dos sistemas modelados. VALLETE (1995) apresenta a utilização da rede de Petri para especificação, análise e implementação de sistemas de controle e monitoração. A representação dos sistemas físicos através das redes de Petri permite a verificação das chamadas “boas propriedades” como: limitação, vivacidade e reversibilidade, que em termos práticos correspondem respectivamente a existência de um número finito de estados, a não existência de auto-travamento (“deadlocks”) e a possibilidade do processo ser repetido infinitamente sem a possibilidade de degradação do processo.

Existem diversos tipos de Redes de Petri que são diferenciados de acordo com o domínio da aplicação no qual eles são utilizados:

- Rede de Petri Condição/Evento (C/E): permite no máximo uma marca nos lugares e um arco entre uma transição e um lugar (Figura D.1).

e transições do tipo discreto e contínuo permitindo a representação dos eventos contínuos e eventos discretos.

- Redes de Petri Diferencial: é uma extensão da Rede de Petri tradicional através da inclusão de um novo lugar (“differential place”) e uma nova transição (“differential transition”) com as regras de evolução associadas. Estes novos elementos permitem a modelagem de forma concorrente de processos dirigidos a eventos e de processos dinâmicos contínuos.

Além disso, para as aplicações práticas existem interpretações específicas para as redes de Petri de modo a uniformizar os modelos e facilitar tanto a sua construção como a sua análise e aplicação como especificação de estratégias de controle.

Para o caso de sistemas produtivos, a metodologia PFS/MFG (MIYAGI, 1996), (ANEXO E) é um exemplo de interpretação das redes de Petri.

ANEXO E - METODOLOGIA PFS/MFG

Esta metodologia consiste na modelagem de um sistema produtivo de forma hierárquica. Inicialmente é criado o modelo conceitual (PFS - “Production Flow Schema”) do sistema; neste caso, são representados as atividades do sistema e os seus inter-relacionamentos. Em seguida, através do detalhamento destas atividades, obtêm-se os grafos MFG (Mark Flow Graph). Tanto o PFS como o MFG são interpretações das redes de Petri próprias para a modelagem de sistemas produtivos.

A dinâmica do sistema é representada através do deslocamento das marcas no modelo em MFG. Nos grafos PFS não existe a representação da dinâmica do sistema.

Os elementos básicos para a criação de um modelo em PFS são (Figura E.1):

- *atividade*: representada pelo par de colchetes e por um nome descritivo da atividade;
- *elemento distribuidor*: representado por uma circunferência;
- *arco*: representado por uma seta entre um elemento distribuidor e uma atividade ou vice-versa e que indica fluxo de materiais e/ou informações.

Figura E.1. Elementos do PFS.

Para a criação de um modelo MFG utilizam-se os elementos apresentados na Figura E.2.

Figura E.2. Elementos do MFG.

Os elementos do grafo PFS podem ser detalhados, de tal forma que são gerados subgrafos totalmente em PFS, subgrafos em MFG ou subgrafos híbridos (PFS/MFG) com alguns elementos em PFS e outros em MFG (Figura E.3).

Esta metodologia permite assim que, através de uma visão macro e conceitual do sistema global, os diferentes subsistemas e suas funções sejam detalhados até o nível de interface com os dispositivos instalados no edifício. A Figura E.3 ilustra uma estrutura hierárquica descrita pelo PFS/MFG.

Figura E.3. Exemplo de um PFS/MFG.

A Figura E.4 apresenta um modelo em MFG (MIYAGI, 1996) da dinâmica do sistema produtivo apresentado na Figura 1.1 (capítulo 1). Neste caso, o evento “chegada de itens” leva a mudança do estado “parado” para o estado “carregando”, o evento “início do processamento” para o estado “processando”, o evento “fim do processamento” para o estado “descarregando” e, finalmente, o evento “fim do descarregamento” que termina o processamento.

Figura E.4. Modelo MFG do sistema produtivo da Figura 1.1.

REFERÊNCIAS BIBLIOGRÁFICAS

- ABRAMSON, A. B. The Intelligent Building System. **Proceedings** of IB/IC Intelligent Buildings Congress. Tel-Aviv, Israel, 1995.
- ANGEL, P.; FRAIGI, L. “Domótica – el hábitat interactivo. Por qué la Domótica?”. Número 1. 1993.
- ANTISAKLIS, P. J. Guest Editorial Hybrid Control Systems: An Introductory Discussion to Special Issue. **IEEE Transactions on Automatic Control** Vol. 43, No. 4, Abril, 1998.
- ARAKAKI, J. **Análise de Sistemas Integrados de Manufatura Baseado no MFG/PFS e Regras de Produção**. Dissertação (Mestrado), Escola Politécnica, Universidade de São Paulo, São Paulo, 1993.
- ARAKAKI, J.; MIYAGI, P. E.; GUSTIN, G. B.; MIYAGI, M. M. Integração de atividades e serviços em edifícios inteligentes – aplicação da metodologia PFS/MFG. **Anais** do Enegep’98, Niterói, RJ, 1998a.
- ARAKAKI, J.; MIYAGI, P. E.; SANTOS FILHO, D. Aplicação da metodologia PFS/MFG na modelagem de Edifícios Inteligentes. **Anais** do II Workshop SintEd sobre Edifícios Inteligentes. Cuba, 1998b.

- ARKIN, H.; PACIUK, M. Service System Integration in Intelligent Buildings. **Proceedings** of IB/IC Intelligent Buildings Congress. Tel-Aviv, Israel. p. 19-30, 1995.
- BENVENISTE A.; LE GUERNIC, P Hybrid Dynamical systems theory and the SIGNAL language. **IEEE Trans. Automatic Control**, vol 34, p. 535-546, Maio, 1990.
- BERNSTEIN, P. Middleware: a model for Distributed System Services. **Communication of the ACM**. p. 86-98, vol. 39, no. 2, February, 1996.
- BOOCH, G. **Object-Oriented Analysis and Design with applications**. The Benjamin/Cummings Publishing Company, 1994.
- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The Unified Modeling Language User Guide**, Addison Wesley, 1999.
- BOYD, D. **Intelligent Buildings**. Editor D. Boyd, British Library Cataloguing in Publication Data, University of Central England, 1994.
- BRENNAN, R. W.; FLETCHER, M.; NORRIE, D. H. Reconfiguration Real-Time Holonic Manufacturing Systems. In: 12th International Workshop on Database and Expert Systems Applications, 2001. **Proceedings**. Munich, Germany. p. 611-615, 2001.
- BRENNAN, R. W.; FLETCHER, M.; NORRIE, D. H. An Agent-Based Approach to Reconfiguration of Real-Time Distributed Control Systems. In: **IEEE Transactions on Robotics and Automation**, vol. 18, no. 4, p. 444-451, August, 2002.
- CAI, X.; VYATKIN, V.; HANISCH, H. M. Design and Implementation of a Prototype Control Systems According to IEC 61499. In: Emerging Technologies and Factory Automation, 2003. **Proceedings**. ETFA '03. IEEE Conference. **Proceedings**. Martin Luther University, p. 269-276, September, 2003.
- CASSANDRAS, C. G. **Discret Event Systems – Modeling and Performance Analysis**. Richard D. Irvin, Inc. and Aksen Associates, Incorporated Publishers. 1993.

- CHAMPAGNAT, R. et al **Petri net based modeling of hybrid systems**. ICMS-NOE ASI'96 Conference Life Cycle Approaches to Production System, Management, Control, Supervision. Toulouse-France, June, 1996.
- CHIRN, J. L.; MCFARLANE, D. C. A Holonic Component-Based Approach to Reconfigurable Manufacturing Control Architecture. In: 11th International Workshop on Database and Expert Systems Applications. **Proceedings**. London UK, p. 219-223, 2000.
- DOUGLASS, B. P. **Real-Time Design Paterns: Robust Scalable Architecture for Real-Time Systems**. Addison Wesley, 2002.
- ELMQVIST, H.; CELLIER F. E.; OTTER, M. Object-Oriented Modeling of Hybrid Systems. **Proceedings** of ESS'93. Netherland, October, 1993.
- FARINES, J. M.; FRAGA, J. S.; OLIVEIRA, R. S. **Sistema em Tempo Real**. In: VII Escola de Computação. Instituto de Matemática e Estatística da Universidade de São Paulo. Julho, 2000.
- FAYAD, M. E.; SCHIMIDT, D. C.; JOHNSON, R. E. **Building Application Frameworks. Object-Oriented Foundations of Framework Design**. Wiley, 1999.
- FLAX, B. M. "Intelligent Buildings". **IEEE Communications Magazine**, p. 24-27, 1991.
- FLETCHER, M.; NORRIE, D. H. Realtime Reconfiguration using an IEC 61499 Operating System. In: 15th International Parallel and Distributed Processing Symposium. **Proceedings**. University of Calgary, p. 985-991, April, 2001.
- FONTOURA, M.; PREE, W.; RUMPE, B. **The UML Profile for Framework Architectures**. Addison Wesley, 2002.
- FOWLER, M.; SCOTT, K. **UML Distilled Second Edition – A Brief Guide to the Standard Object modeling Language**. Third edition. Addison Wesley, 2003.
- FUJIMOTO, Y. Fault-Tolerant Configuration of Distributed Discrete Controllers. In: **IEEE Transactions on Industrial Electronics**, vol. 50, no. 1, p. 475-486, February, 2003.

- GAMMA, E. et al. **Design Patterns. Elements of Reusable Object-Oriented Software**. Addison Wesley, 1994.
- GOMES, L. “Edifícios Inteligentes: um conceito potencialmente mobilizador”. In: Capítulo 6. Tese de doutoramento em Engenharia Electrotécnica, Especialidade de Sistemas Digitais, Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia, Lisboa – Portugal, p. 247-289, 1998.
- GOMES, L.; STEIGER-GARÇÃO, A. “Domots are coming ! or How to manage building automation in balanced way ?”. **Anais** do BASYS’96 – 2nd IEEE/ECLA/IFIP International Conference on Architectures and Design Methods for the Balanced Automation Systems, Lisboa, Portugal, 1996.
- GROOVER, M. P. **Automation, Production, System and Computer-Integrated Manufacturing**, Second Edition, Prentice Hall, 2000.
- HECK, S. B.; WILLS, L.M; VACHTSEVANOS, G. J. Software Technology for Implementing Reusable, Distributed Control Systems. **IEEE Control Systems Magazine** Vol. 23, No. 3, p. 21-36, February, 2003.
- HO, Y. C. Introduction to Special Issue on Dynamics of Discrete Event Systems. **Proceedings** of IEEE, v. 77, p. 3-6, 1989.
- ISO/IEC 9126. Information Technology – Software Product Quality. 1998.
- ITO, Y. A diserable production structure looking toward the 21st century – Antropocentric Intelligence-Based Manufacturing. In: XI COBEM. **Anais**, p. 23-32, São Paulo, Brasil, 1991.
- JENNINGS, N. R. e BUSSMANN, S. Agent-Based Control Systems: Why are they suited to engineering complex systems?. **IEEE Control Systems Magazine** Vol. 23, No. 3, p. 61-73, June, 2003.
- JUGEND, D. Automação Predial – Evolução e Comparativo. **Anais** do XI CBA – Congresso Brasileiro de Automação/ SBA – Sociedade Brasileira de Automática, São Paulo, p. 179-196, 1996.

- KAMINKA, G. A.; TAMBE, M. What is Wrong With Us ? Improving Robustness Through Social Diagnosis. **Proceedings** do 15o. National Conference on Artificial Intelligence, 1998.
- KLEIN, M.; DELLAROCAS, C. Exception Handling in Agent Systems. **Proceedings** of the 3th Annual Conference on Autonomous Agents, Washington, USA, p. 62-68. 1999.
- KOWALEWSKI, S. A Case Study in Tool-Aided Analysis of Discretely Controlled Continuous Systems: the Two Tanks Problem. **Proceedings** of the 5th Int. Workshop On Hybrid Systems, Notre Dame, USA, 1997.
- KUMAR, S.; COHEN, P. R. Towards a Fault Tolerant Multi-Agent System Architecture. p. 459-466. **Proceedings**. Fourth International Conference on Autonomous Agents. Barcelona, Spain, 2000.
- LADDAGA, R. Active Software. In: 1th International Workshop on Self-Adaptative Software (IWSAS2000). Oxford, UK, Springer-Verlag, 2000.
- LARMAN, C. **Applying UML end Patterns. An Introduction to Object-Oriented Analysis and Design and the Unified Process**. 2a. ed. Prentice Hall, 2002.
- LEWIS, R. Modelling control systems using IEC 61499, In: **IEE Control Engineering series 59**, 2001.
- LEWIS, R. Programming industrial control systems using IEC 61131-3, In: **IEE Control Engineering**, 1996.
- MCCLELLAND, S. **Intelligent Building; An IFS Executive Briefing**. Spring Verlag, Blenheim OnLine, Bedford England, 1988.
- MIYAGI, P. E. **Control System Design, Programming and Implementation for Discret Event Production Systems by using Mark Flow Graph** Doctor Thesis, Tokyo Institute of Technology, Tokyo, Japan, 1988.
- MIYAGI, P. E.; BARRETO, M. R. P.; SILVA, J. R. **Domótica: Controle e Automação**. VI EBAI (Escuela Brasileño Argentina de Informática), Embalse do Rio Tercero, Argentina, 1993.

- MIYAGI, P. E. **Controle Programável – fundamentos do controle de sistema a eventos discretos**. Editora Edgard Blücher, São Paulo, 1996.
- MIYAGI, P. E., ARAKAKI, J. e SILVA, J. R. Ambiente para o Desenvolvimento e Teste de técnicas de integração de sistemas em edifícios inteligentes. **Anais** do I Workshop SintEd sobre Edifícios Inteligentes. Colombia, pp.13-22, 1997a.
- MIYAGI, P. E., ARAKAKI, J.; SANTOS FILHO, D. J. Projeto de Sistemas de Controle Programáveis. **Anais** do Congresso Petrobrás de Informática e Telecomunicações. São Paulo, 1997b.
- MIYAGI, P. E., ARAKAKI, J.; GUSTIN, G. B.; MIYAGI, M. M.; KISIL, M. Aplicação do PFS/MFG na modelagem de atividades e serviços em Edifícios Inteligentes. **Anais** da 6a. Reunião Anual da Sociedade Brasileira de Pesquisadores Nikkeis. Ilha Solteira, SP, 1998a.
- MIYAGI, P. E.; MIYAGI, M.M.; SANTOS FILHO, D. J.; ARAKAKI, J. Aplicação de redes de Petri para análise de Sistemas de Saúde em ambiente de Edifícios Inteligentes. **Anais** do II Workshop SintEd sobre Edifícios Inteligentes. Varadero, Cuba, 1998b.
- MOORE, M. L. et al. Complex Control System Design and Implementation using the NIST-RCS software library. **IEEE Control Systems Magazine**. Vol. 19, No. 6, p. 12-28, December, 1999.
- MURATA, T. Petri Nets: Properties, Analysis and Applications. **Proceedings** of the IEEE, Vol. 77, No. 4, p. 541-580, Abril, 1989.
- NAKAMOTO, F. Y. **Sistematização do projeto do controle de sistemas produtivos**. Dissertação (Mestrado), Escola Politécnica, Universidade de São Paulo, São Paulo, 2002.
- ODELL, J. Objects and Agents Compared. **Journal of Object Technology**, vol. 1, no. 1, p. 41-53, Maio-Junho, 2002.
- OESTEREICH, B. **Development Software with UML – Object-Oriented Analysis and Design in Practice**. Addison Wesley, 1999.

- ORFALI, R.; HARKEY, D. **Client/Server Programming with JAVA and CORBA**, 2nd. Edition, John Wiley and Sons, inc. 1998.
- PETERSON, J. L. **Petri Net Theory and the Modeling of Systems**. Prentice Hall, Englewood Cliffs, N. J., 1981.
- PUTMAN, J. R. **Architecting with RM-ODP**. Prentice Hal PTR, 2001.
- QNX Software Systems. Real Time Operating Systems. Disponível em: <http://www.qnx.com/products/rtos/>. Acesso em: 22 Out. 2004.
- RAMADGE, P. J.; WHONHAM, W. M. The Control of Discrete Event Systems. **Proceedings** of IEEE, v.77, no. 4, 1989.
- REILLY, D.; TALEB-BENDIAB, A.; LAWS, A.; BADR, N. An Instrumentation and Control-Based Approach for Distributed Application Management and Adaptation. In: The first workshop on Self-healing systems. **Proceedings**. Charleston, South Carolina, USA, p. 61-66, 2002.
- REISIG, W. **Petri Nets: An Introduction**. Spring-Verlag, Berlin Heidelberg, 1985.
- REISIG, W. **A Primer in Petri Nets Desing**. Spring-Verlag, Berlin Heidelberg, 1992.
- RIASCOS, **Metodologia para Detecção e Tratamento de Falhas em Sistemas de Manufatura através de Redes de Petri**. Tese (Doutorado), Escola Politécnica, Universidade de São Paulo, São Paulo, 2002.
- RICH, E.; KNIGHT, K. **Artificial Intelligence**. McGrawHill, 1991.
- ROWE, N. C. **Artificial Intelligence Through Prolog**, Prentice Hall, 1988.
- RUMBAUGH, J. et al. **Modelagem e Projetos Baseados em Objetos**. Tradução autorizada. Editora Campus, 1994.
- SANTOS FILHO, D. J. **Proposta do Mark Flow Graph Estendido para a modelagem e Controle de Sistemas integrados de Manufatura**. Dissertação (Mestrado), Escola Politécnica, Universidade de São Paulo, São Paulo, 1993.

- SANTOS FILHO, D. J. **Controle de Sistemas Antropocêntricos de Produção Baseado em Redes de Petri Interpretadas**. Tese (Doutorado), Escola Politécnica, Universidade de São Paulo, São Paulo, 1998.
- SANTOS FILHO, D. J. **Aspectos do projeto do controle de sistemas produtivos**. Tese (Livre Docência), Escola Politécnica, Universidade de São Paulo, São Paulo, 2000.
- SANZ, R.; ARZÉN, K. E. Trends in Software and Control. **IEEE Control Systems Magazine**. Vol. 23, No. 3, p. 61-73, June, 2003.
- SCARLETT, J.; BRENNAN, R. W.; NORRIE, D. H.. An Interface to Support Real-Time Distributed Control. In: IEEE Conference on System, Man Cybernetics. **Proceedings**. pp.636-641, 2003.
- SILVA, V.; LUCENA C. MAS-ML: A Multi-Agent System Modeling Language. In: OOPSLA '03. **Anais**. Anaheim, Califórnia USA, p. 126-127, 2003.
- STANKOVIC, J. Misconceptions About Real Time Computing – A Serious Problem for Next-Generation Systems. **IEEE Computer Magazine**. p. 11-18. October, 1988.
- STUBBINGS, M. **Intelligent Building; An IFS Executive Briefing**. Spring Verlag, Blenheim OnLine, Bedford England, 1988.
- TANENBAUM, V. S. **Distributed Systems – Principles and Paradigms**. Prentice Hall, 2002.
- VALLETE, R. Petri nets for control e monitoring: specification, verification, implementation. **Anais** do Workshop on Analysis and Design of Event – Driven Operation in Process Systems, Imperial College, London, Abril, 1995.
- VILLANI, E.; MIYAGI, P. E.; SANTOS FILHO, D. J.; MARUYAMA, N.; ARAKAKI, J. Modelagem do Sistema de Controle das Condições de Conforto Térmico em Edifícios Inteligentes . **Anais** do Congresso Brasileiro de Engenharia Mecânica, Águas de Lindóia, 1999.
- WADA, H.; OKADA, S. An autonomous agent approach for manufacturing execution control systems, Integrated Computer-Aided Engineering, p. 251-262, 2002.

- WANG, S.; SHIN, K. G. Constructing Reconfigurable Software for Machine Control Systems. In: **IEEE Transactions on Robotics and Automation**, vol. 18, no. 4, p. 475-486, August, 2002.
- WILLS, L.; KANNAN, S.; SANDER, S.; GULER, M.; HECK, B.; PRASAD, J.V.R.; SCHRAGE, D.; VACHTSEVANOS, G. An Open Platform for Reconfigurable Control. **IEEE Control Systems Magazine**. p. 49-64, June, 2001.
- WINSTON, P. H. **Inteligência Artificial**. Addison Wesley, 1992.
- ZHANG, X.; NORRIE, D. H.; BRENNAN, R. W.; XU, Y. A Multi-level Reconfiguration Control for Holonic PLC. Calgary, Canada, p. 1762-1767, 2000.