PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

CARLOS ALBERTO VICARI

O uso da simulação no suporte a implementação da Estrutura de Controle Supervisório

CURITIBA
2008

CARLOS ALBERTO VICARI

O uso da simulação no suporte a implementação da Estrutura de Controle Supervisório

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

Área de concentração: Automação e Controle de Sistemas

Orientador: Prof. Dr. Eduardo Alves Portela Santos
Co-orientador: Prof. Dr. Marco Antônio Busetti de Paula

CURITIBA
2008

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central

V628u
2008

Vicari, Carlos Alberto
O uso da simulação no suporte a implementação da estrutura de controle supervisório / Carlos Alberto Vicari ; orientador, Eduardo Alves Portela Santos ; co-orientador, Marco Antônio Busetti de Paula . -- 2008.
xi, 92 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2008
Bibliografia: f. 74-78

1. Processos de fabricação - Automação. 2. Controle de produção. 3. Produtos industrializados. 4. Simulação (Computadores). I. Santos, Eduardo Alves Portela. II. Paula, Marco Antônio Busetti de. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Engenharia de Produção e Sistemas. IV. Título.

CDD 20. ed. – 670.427

“O fato de ser brasileiro só me enche de orgulho”

Ayrton Senna

AGRADECIMENTOS

A minha esposa Eliane Costa Vicari e minha filha Maria Eduarda Vicari, pelo incentivo, apoio no desenvolvimento deste trabalho e compreensão nos momentos em que ficamos distantes.

Aos meus pais, Aquiles e Maria, agradeço a força e educação que me fizerem iniciar e terminar esse trabalho com dedicação e qualidade.

A meus professores orientadores, Prof. Eduardo Portela e Prof. Marco Busetti, pela confiança em mim depositada, auxiliando-me no desenvolvimento e na compreensão do que foi necessário para a conclusão deste trabalho.

A meus colegas de mestrado, em especial a Ricardo Diogo, pelo auxílio no entendimento dos conceitos necessários para a conclusão deste trabalho; aos professores e colaboradores do programa de Pós-Graduação em Engenharia de Produção e Sistemas pelos diversos subsídios fornecidos durante meus estudos e trabalhos.

SUMÁRIO

1 INTRODUÇÃO 1
1.1 Objetivos 5
1.2 Metodologia de trabalho adotada 6
1.3 Estruturação 7
2 FUNDAMENTAÇÃO TEÓRICA 9
2.1 Sistemas a Eventos Discretos 9
2.2 Controle supervisório de sistemas a eventos discretos 12
2.3 Controle modular local 16
2.4 Modelagem de sistemas compostos 18
2.5 Implementação de supervisores locais 20
2.6 Estrutura de Controle proposta por Vieira (2004) 21
2.7 Ciclo de Desenvolvimento de sistemas automatizados e integrados 26
2.8 Simulação 28
2.9 Hardware-in-the-Loop 30
2.10 Considerações Finais 32
3 DESENVOLVIMENTO DE UMA SISTEMÁTICA DE SIMULAÇÃO 34
3.1 Ambiente de implementação 34
3.2 Ambiente de Simulação 37
3.3 Interface 41
4 ETAPA DE IMPLEMENTAÇÃO 43
4.1 Fase de simulação 43
4.2 Fase de simulação + CCT 52
4.3 Fase de execução 57
5 EXEMPLO DE APLICAÇÃO 58
5.1 Definição do ambiente de estudo de caso 58
5.2 Levantamento de informações do ambiente de simulação 61
5.2.1 Escolha das tecnologias 63
5.3 Integração do ambiente de controle e comunicação com o ambiente dinâmico 64
5.4 Validação da sistemática 67
6 CONCLUSÃO 71

REFERÊNCIAS.....74
APÊNDICE.....79

LISTA DE FIGURAS

Figura 1.1 – Estruturação da dissertação....8
Figura 2.1 – Evolução típica de um Sistema a Eventos Discretos. 10
Figura 2.2 – Sistemas a Variáveis Contínuas e Sistemas a Eventos Discretos. 12
Figura 2.3– Acoplamento da planta e supervisor no modelo RW (RAMADGE e WONHAM, 1989). 13
Figura 2.4- Alfabeto de eventos de um sistema composto....19
Figura 2.5– Estrutura básica do sistema de controle (Queiroz et al., 2001). 21
Figura 2.6 – Estrutura de controle 22
Figura 2.7 – Ciclo de desenvolvimento para um FMS 26
Figura 2.8 – Ciclo de desenvolvimento para simulação com HiL 31
Figura 3.1 – Etapa de implementação e o Sistema de Controle – Diogo et al (2008) 35
Figura 3.2 –Detalhamento do ambiente de simulação 38
Figura 3.3 – Níveis do ambiente de simulação....39
Figura 3.4 – Evolução da simulação, Diogo et al (2007). 41
Figura 4.1– *Layout* do arquivo de controle de comunicação 45
Figura 4.2 – Função de definição e inicialização das variáveis 47
Figura 4.3 – Função leitura e verificação de gravação de comando 47
Figura 4.4- Função leitura dos comandos 48
Figura 4.5- Função define *ackread*....48
Figura 4.6 – Fluxo de controle de troca dos comandos....49
Figura 4.7 – Função leitura da resposta....50
Figura 4.8 – Função incremento da resposta....50
Figura 4.9 – Função gravação da resposta....50
Figura 4.10 – Fluxo de controle de troca das respostas....51
Figura 4.11 – Ambiente simulado + ambiente real 53
Figura 4.12 – Arquivo de controle de comunicação com OP executada ambiente real e virtual....53
Figura 4.13 – Função gravação da resposta 54
Figura 4.14 – Fluxo de controle de comandos para o Ambiente de Simulação + CCT 55

Figura 4.15 – Fluxo de controle de respostas para o Ambiente de Simulação + CCT 56
Figura 5.1 – Sistema de manufatura de três máquinas.....58
Figura 5.2 – Geradores das máquinas M1, M2 e M3 59
Figura 5.3 – Especificações de controle E1 e E2.....59
Figura 5.4 – Supervisores modulares para o sistema de manufatura 60
Figura 5.5 – Diagramas funcionais dos Procedimentos Operacionais do sistema de manufatura 60
Figura 5.6 – *Layout* do arquivo de controle de comunicação (completo.xls).....63
Ffigura 5.7 – Nível de procedimento operacional implementado.....65
Figura 5.8 – Detalhamento do Processo 3 65
Figura 5.9 – Nível de sistemas virtuais.....66
Figura 5.10 – Implementação do nível de interface.....66
Figura 5.11 – Interface com o usuário para a aplicação do AD para o exemplo de três máquinas.....68
Figura 5.12 – Arquivo de controle para simulação 69
Figura 5.13 – Ambiente de simulação 69
Figura 5.14 – Interface com o usuário para a aplicação do AD para o exemplo de três máquinas, sendo uma implementada no ambiente real. 70

LISTA DE QUADROS E TABELAS

Tabela 4.1– Matriz de subsistemas e procedimentos operacionais 43
Tabela 4.2– Matriz de Comandos e Repostas 44
Tabela 4.3 – Relacionamento entre a matriz de comunicação e as variáveis e apontamentos nos *software* 45
Tabela 5.1 – Matriz subsistemas e seus procedimentos operacionais.....62
Tabela 5.2 – Matriz de comandos e repostas.....62
Tabela 5.3 – Matriz de controle 62
Tabela 5.4 – Sistemas a serem controlados 62
Tabela 5.5 - Relacionamento entre a matriz de comunicação e as variáveis e apontamentos nos *software.* 63

LISTA DE ABREVIATURAS

ACS – Arquitetura de Controle Supervisório

AD – Aplicação Dinâmica

AMS – *Agile Manufature System*

BMC – Biblioteca de Modelos de Comunicação

CCMs – *Cellular Manufacturin Systems*

CCT – *Control Communication Technologies*

CLP – Controlador Lógico Programável

CML – Controle Modular Local

DDE – *Dynamic Data Exchange*

DMSs – *Dedicated Manufacturing Systems*

E/S – Entrada/Saída

FMS – *Flexible Manufacturing System*

HiL – *Hardware-in-the-Loop*

ISO – *International Organization for Standardization*

MPS – *Manufacturing system prototype*

MS – *Manufacturing Systems* - Sistema de Manufatura

PC – Personal Computer

PO – Procedimentos Operacionais

RSP – Representação por sistema-produto

SCADA – *Supervisory Control And Data Acquisition*

SED – Sistemas a Eventos Discretos

SM – Supervisores Modulares

SP – Sistema-produto

SF – Sistema Físico

TCC – Tecnologias de Comunicação e Controle

TCS – Teoria de Controle Supervisório

VLSI –*very large scale intragration*

VPS – *Virtual Production Systems*

RESUMO

O crescimento da competitividade nas empresas força-as a reduzir cada vez mais o tempo de lançamento de seus produtos. Como conseqüência disso, um esforço, tanto por parte das empresas quanto por parte da academia, para fornecer técnicas e ferramentas para a rápida reconfiguração do sistema de controle em sistemas de manufatura está em vigor. Um sistema de manufatura é um exemplo clássico de um Sistema a Eventos Discretos (SED) porque sua dinâmica evolui de acordo com a ocorrência de eventos discretos ao longo do tempo. Nesse contexto, métodos formais para a modelagem, análise e síntese de SED, como a Teoria de Controle Supervisório (TCS), têm contribuído com a solução de alguns problemas. Porém, existem ainda limitações a este formalismo, como a centralização do agente supervisor. Extensões da TCS como o Controle Modular Local (CML) são alternativas para descentralizar o agente supervisor e propiciar a reconfiguração do sistema de manufatura de forma mais rápida. A partir daí, verificou-se a necessidade de se elaborar um ciclo de desenvolvimento composto por etapas. A primeira delas é a modelagem de SED; a segunda, a síntese dos supervisores modulares e uma terceira, que é a implementação. Nesta última, o modelo teórico é traduzido para uma linguagem compatível à plataforma industrial, mostrando a necessidade de haver métodos de validação antes de o equipamento ser conectado ao sistema de manufatura. Um método de validação possível consiste na simulação da estrutura de controle e de sua inserção gradativa ao sistema real. Este trabalho propõe o uso da Simulação como ferramenta de suporte à implementação da estrutura de controle supervisório. Por meio dessa simulação, a migração gradativa de subsistemas físicos simulados para subsistemas físicos reais é possível, com o uso da técnica conhecida como *Hardware-in-the-Loop* (HiL). Essa técnica pressupõe a validação da estrutura de controle obtida por meio da simulação e, gradativamente, permite a inserção dos controladores reais da planta física. Ao longo da validação, os subsistemas físicos anteriormente simulados são inseridos ao sistema de manufatura. Finalmente, um exemplo de aplicação a um sistema de manufatura é proposto, validando, assim, a abordagem apresentada neste trabalho.

Palavras-chave: Sistemas de Manufatura, Sistemas a Eventos Discretos, Teoria de Controle Supervisório, Controle Modular Local, Controlador Lógico Programável, Integração de Sistemas, Simulação, *Hardware-in-the-Loop*.

ABSTRACT

The growth of competitiveness in enterprises has forced them to reduce more the releasing time of their products. Consequently, the enterprise and academy's effort in providing techniques and tools for the reconfiguration of its systems is in force. A manufacturing system is a classical example of a Discrete-Event System (DES), for its dynamics develops according to the occurrence of discrete events over time. In this context, formal methods of DES modeling, analysis and synthesis, such as Supervisory Control Theory (SCT), have contributed for the solution of some problems, although there are still some limitations to this formalism, for instance, the centralization of the supervisor agent. Extensions of SCT, such as Local Modular Control (LMC), are alternatives in order to decentralize the supervisor agent and provide the reconfiguration of the manufacturing system more quickly. From this point, there is the requirement of elaborating a development sequence, made of steps. The first one is the DES modeling. The second one is the modular supervisors' synthesis, and the third one is the implementation itself. In the latter, the theoretical model is translated to a language suited to the industrial platform, which presents the need of having validation methods before connecting the equipment to the manufacturing system. A possible validation method consists in simulating the controlling structure and inserting it gradually to the real system. This work assumes using the simulation as a support tool to the implementation of the supervisory control structure. By this modeling, the physical subsystems gradual migration to real physical subsystems is made possible, by using the technique known as Hardware-in-the-Loop (HiL), which assumes the validation of the control structure, achieved by the simulation and, gradually allows the insertion of the real controllers of the physical plant. Along the validation, the physical subsystems previously simulated are placed into the manufacturing system. Finally, an application example to the manufacturing system is projected. This way, the approach presented in this work is validated.

Keywords: Manufacturing systems, Discrete-Event Systems, supervisory control theory, systems integration, simulation, Hardware-in-the-Loop, SCADA.

1 INTRODUÇÃO

Com o advento da globalização, as empresas são levadas cada vez mais a aprimorar seus produtos e a ter respostas rápidas para sua adequação às novas necessidades apresentadas. Com isso, elas têm buscado soluções tecnológicas para atender de maneira mais eficiente e com baixo custo às constantes modificações em suas plantas e linhas de produção.

Os sistemas de manufatura tendem a evoluir de forma mais rápida e, por isso, para garantir a agilidade, funcionalidade e baixo custo, cada vez mais estudos na área de simulação vêm sendo desenvolvidos (Holst et al, 2001).

Esses estudos têm como principal objetivo elaborar procedimentos que facilitem a implementação de tecnologias que auxiliam o desenvolvimento de uma manufatura ágil.

Tendo em vista a perspectiva apresentada, o estudo da teoria de controle supervisório, simulação, engenharia virtual, e manufatura ágil vêm de encontro à necessidade de se criar soluções para atender a essas novas necessidades e à demanda das empresas.

Os conceitos de TCS, HiL, Simulação e Manufatura Ágil abordados nesse trabalho integram, de forma conjunta, uma solução de implementação de uma sistemática para simular e implementar modelos de plantas baseados na TCS. Será usado como base para a implementação dessa sistemática, o ciclo de desenvolvimento de produto proposto por Busetti e Santos (2006), com o objetivo de complementar os trabalhos já iniciados por eles, sendo uma proposta de solução nesta área.

A Teoria de Controle Supervisório (TCS) foi desenvolvida com o objetivo de fornecer uma metodologia formal para a síntese automática de controladores para Sistemas a Eventos Discretos (Ramadge e Wonham, 1989). A partir do trabalho desenvolvido por Ramadge e Wonham (1989), outros foram desenvolvidos com o intuito de complementá-lo, expandi-lo e divulgá-lo (Fabian e Hellgren, 1998; Hellgren et al, 2002; Leduc, 1996; Queiroz e Cury, 2002; Vieira et al, 2006; Vieira, 2008).

Dentre os trabalhos estudados, percebe-se a que TCS está evoluindo em termos de modelos teóricos; porém, existe a necessidade de implementá-los fisicamente. Essa implementação física apresenta-se como o maior desafio encontrado, e, além de implantar os códigos das ações de controle, é necessário elaborar uma sistemática que simule e valide a TCS.

A distribuição da estrutura de controle é um dos itens fundamentais quando relacionado aos fatores de limitação de plataformas (memória disponível, quantidade de entradas e saídas) e a modularidade do sistema físico, de acordo com Viera e Cury (2004). Além disso, a distribuição do sistema físico de controle admite uma melhor organização dos itens da planta, permitindo alterações futuras, manutenções em itens parciais e, nesse caso, a execução parcial dela, a fim de identificar seu funcionamento individual ou total.

Conforme proposto por Gertosio et al (2000), é possível usar uma abordagem monolítica para a distribuição física dos supervisores. Essa proposta indica uma descentralização das estações de controle e um supervisor é atrelado a cada célula de manufatura. Desta forma, quando há alteração em alguma célula ou ela é retirada do processo, é necessário somente alterar ou excluir o respectivo supervisor. Essa proposta porém, gera uma divergência, pois como esses supervisores trocam informações de controle com o gerenciador da produção, exclusivamente, e não com os demais supervisores, isso implica em uma estrutura de controle hierárquica, na qual há centralização do controle da produção em um gerenciador devidamente centralizado. Isso torna a estrutura de controle monolítica e caso haja alterações na planta deve haver a necessidade de se alterar o *software* de gerenciamento centralizado, o que não permite modularidade na estrutura de controle.

Alguns trabalhos foram desenvolvidos tendo em vista a modularidade, para propor uma solução para o problema apresentado, dentre os quais se destacam Queiroz e Cury (2002), Vieira et al (2006) e Vieira (2007)

Complementando esse contexto, identifica-se a Engenharia Virtual e a Manufatura Virtual como novas tecnologias de manufatura ágil capazes de modelar sistemas virtuais com base em processos reais e achar possíveis soluções para um processo, complementando-os com informações e métodos mais aprimorados (Moore et al, 2003). A utilização deste conceito, portanto, irá completar

a lacuna entre as necessidades para a implementação TCS e os recursos tecnológicos disponíveis atualmente.

Busseti & Portela (2008) propõem o ciclo de desenvolvimento de processos reconfiguráveis, que implementa uma metodologia que é aplicada a sistemas automatizados e é integrada com a produção, em um desenvolvimento cíclico de três etapas – modelagem, síntese e implementação – com funções bem definidas. Na etapa de modelagem é especificado o sistema-produto que usa autômatos assíncronos, modelados isoladamente somente os eventos relevantes. Na etapa de síntese são calculadas as menores linguagens controladas e gerar os supervisores que controlam o modelo. E por fim, a etapa de implementação gera o código, baseado nos supervisores e fornecido pela etapa de síntese, que é utilizado em um simulador no qual, posteriormente, serão gradativamente incorporados os subsistemas reais até sua substituição completa. Desta forma, o clico completa-se e os resultados obtidos no simulador são analisados e disponibilizados por meio de uma biblioteca – a etapa de modelagem – para efetuar as devidas correções no modelo se necessário. A metodologia *Hardware in the Loop* – HiL de simulação – é uma forma de abordar esse assunto.

Segundo Boyd e Theyyunni (1999), a simulação é uma maneira rápida de testar o protótipo e a ela podem ser aplicados os sistemas físicos em tempo real, pois admitem aperfeiçoar a planta de manufatura. Para isso, Boyd e Theyyunni (1999) propõem o uso do HiL em sistemas de simulação em tempo real. A partir disso, podem-se obter dados para alimentar uma cadeia de manufatura ágil, utilizando como base um simulador comercial e um *driver* de Controlador Lógico Programável (CLP) como aplicativo de controle entre o ambiente real e o simulador. Desta forma podem-se migrar gradativamente subsistemas simulados para o ambiente real.

Em Stoeppler et al (2005), verificam-se os seguintes motivos para o uso de HiL em sistemas de simulação em tempo real:

a. podem não haver maneiras físicas viáveis para construção de protótipos;
b. o controle do ambiente de teste real pode ser muito difícil;
c. questões de segurança podem tornar os testes de execução impossíveis no sistema;
d. considerações econômicas ditam o uso da simulação;
e. o tempo de *setup* reduzido.

Nos trabalhos apresentados por Queiroz e Cury (2002), Vieira et al (2006) e Vieira (2008), a estrutura de controle é amplamente detalhada, e são propostas soluções operacionais para problemas como a controlabilidade e síntese dos modelos da estrutura de controle. Porém, esses autores limitam-se a descrever a estrutura de controle e não prevêem uma sistemática de implantação dessa estrutura de controle como solução para os problemas apresentados.

Ainda, questões como validação dos modelos, testes, detecção de erros e análise de desempenho dos modelos não são descritas, fatos que mostram a necessidade de ampliar esses trabalhos propondo uma solução para a implementação e validação da estrutura de controle em ambientes simulados e em ambientes reais.

Para P.R, Moore et al (2003), a manufatura virtual usada para desenvolvimento, testes, programação e validação atende às necessidades do mercado globalizado, sendo possível melhorar a qualidade dos produtos, diminuir os custos de produção, e atender à constante diminuição de seus tempos de vida.

Em Lin Li (2006), a Teoria de Controle Supervisório, quando comparada às Redes de Petri, provê os mesmos recursos de modelagem e é mais eficiente no item controle. Também para Lin Li (2006), com o desenvolvimento de tecnologias computacionais, a previsão de uso de Sistemas de Produtos Virtuais (VPSs) no futuro encoraja o estudo desses recursos no presente.

Como a metodologia proposta por Busseti e Portela (2008) supre essa necessidade, a possibilidade de se complementar essa metodologia com uma sistemática capaz de simular um Sistema Produto de forma completa, mostra-se como uma solução viável de implementação na manufatura ágil.

Um fato motivador é que, cada vez mais as empresas necessitam reconfigurar seus sistemas produtivos ou programar novas linhas de produção em um tempo menor. Porém, em alguns casos, alguns subsistemas já modelados e simulados são passíveis de implementação no sistema físico real sem que o modelo inteiro seja implementado fisicamente. Desta forma, é possível analisar o resultado do sistema físico com o sistema simulado em tempo real e, assim, ganhar agilidade caso seja necessário reconfigurar algum modelo.

1.1 Objetivos

Essa dissertação tem, como objetivo geral, criar um ambiente de suporte a simulação e implementação da Arquitetura de Controle Supervisório, com uso da metodologia *Hardware-in-the-Loop* no contexto da manufatura ágil de Busseti e Portela (2008) integrando um ambiente de simulação a uma estrutura de controle implementada fisicamente através de Controladores Lógicos Programáveis. Esse ambiente também se utiliza do Ambiente Dinâmico, definido em Diogo et AL (2008), como ferramenta de suporte à simulação.

No uso da estrutura de controle, existe o transporte de informações entre as várias camadas da estrutura de forma hierárquica, e também há uma conexão entre a estrutura de controle e o sistema físico. Esse trabalho propõe uma sistemática para a criação de uma conexão entre as camadas de Procedimentos Operacionais e Sistema-produto da estrutura de controle, que serão simulados com o sistema físico, possibilitando também uma forma de migração dos Procedimentos Simulados para Procedimentos Operacionais reais de forma gradativa, a fim de validar os modelos e a estrutura de controle proposta por Queiroz e Cury (2002).

Como prática, propõe-se a implementação de um estudo de caso com o uso do *software* Arena para efetuar as simulações, inicialmente, dos Procedimentos Operacionais e do Sistema Físico. Gradativamente, estes devem ser substituídos por Procedimentos Operacionais e Sistemas Físicos reais. Para auxiliar essa implementação será usado o *software* Elipse como um Ambiente Dinâmico, conforme proposto por Diogo et al (2008), para controlar as comunicações entre os sistemas reais e os sistemas virtuais. Essa sistemática usa conceitos de metodologia HiL como base para o controle dos modelos e das técnicas para simulação.

Para isso, é necessário realizar as ações descritas a seguir como objetivos específicos:

- conceber um ambiente de simulação para a implementação e validação da Arquitetura de Controle Supervisório;
- integrar o ambiente à Arquitetura de Controle Supervisório por meio de um ambiente dinâmico;
- realizar experimentos para testes do ambiente de simulação proposto;

• aproximar o tempo de execução da simulação ao tempo de execução da planta física real utilizando conceitos da metodologia *Hardware-in-the-Loop.*

O resultado da sistemática irá proporcionar informações que subsidiarão estratégias de produção para que possam ser tomadas decisões de correção e otimização dos processos analisados.

1.2 Metodologia de trabalho adotada

Como metodologia de desenvolvimento escolhida para esse trabalho, é proposto inicialmente uma revisão bibliográfica, que visa a suprimir de recursos e subsídios teóricos o desenvolvimento do ambiente proposto, passar por uma fase de desenvolvimento experimental, na qual são inicialmente modelados os subsistemas e testadas as tecnologias, documentar os procedimentos necessários para novas implementações e por fim, efetuar a implementação completa de um sistema para .validar o conteúdo teórico.

Uma vez que a metodologia de experimentação para a elaboração deste ambiente norteia as atividades deste trabalho, o mesmo irá consistir em testar as diversas formas de implementação de simulação, teoria de controle supervisório, tecnologias de comunicação e outros aspectos tecnológicos envolvidos, até atingir-se uma solução capaz de possibilitar a implementação do ambiente proposto, de forma que o objetivo geral deste trabalho seja atingido com o mínimo de restrições possíveis.

Especificamente, em um momento inicial, são verificados os requisitos tecnológicos envolvidos e as condições tecnológicas para que a sistemática proposta seja implementada. Se houver viabilidade técnica, são analisados os conteúdo da Teoria de Controle Supervisório, validando assim sua estrutura nesta proposta. Como essa análise é baseada na literatura técnica da área, serão buscados recursos adicionais, como as tecnologias de simulação de processos de manufatura e a metodologia de *hardware-in-the-loop*, para desenvolver a solução da problemática e validar, através de implementações e testes , os recursos já desenvolvidos e utilizados por outros autores.

Depois de uma fase inicial de análise das tecnologias e de validação da possibilidade da implementação, é elaborada uma sistemática de desenvolvimento, a qual é validada com a implementação real de um exemplo de sistema de manufatura. Esse exemplo compõe-se de três recursos físicos e três procedimentos operacionais, que englobam suficientemente os requisitos para validar a implementação desta sistemática.

1.3 Estruturação

No capítulo 1 são observados os conceitos introdutórios, em que são observados o objetivo da dissertação, a contextualização do problema e a motivação descrita.

O capítulo 2 descreve qual o tipo de metodologia de pesquisa é usada e de que forma são analisados os resultados.

O capítulo 3 revisa os fundamentos teóricos necessários para o desenvolvimento e a implementação da metodologia sugerida. Os principais tópicos analisados são a simulação, o Controle Supervisório, a Manufatura Ágil e *Hardware-in-the-Loop*. Esses tópicos servem como base teórica para o desenvolvimento da sistemática e para sua implementação.

No capítulo 4 são apresentados os desenvolvimentos e passos da sistemática proposta e no capítulo 5 é apresentada a implementação da sistemática em um estudo de caso.

O capítulo 6 conclui os resultados obtidos na implementação da metodologia proposta nessa dissertação e sugere possíveis melhorias para novos trabalhos e supostas dificuldades encontradas.

A Figura 1.1 mostra, de forma gráfica, as relações entre os capítulos e a seqüência estruturada para o desenvolvimento dessa dissertação.

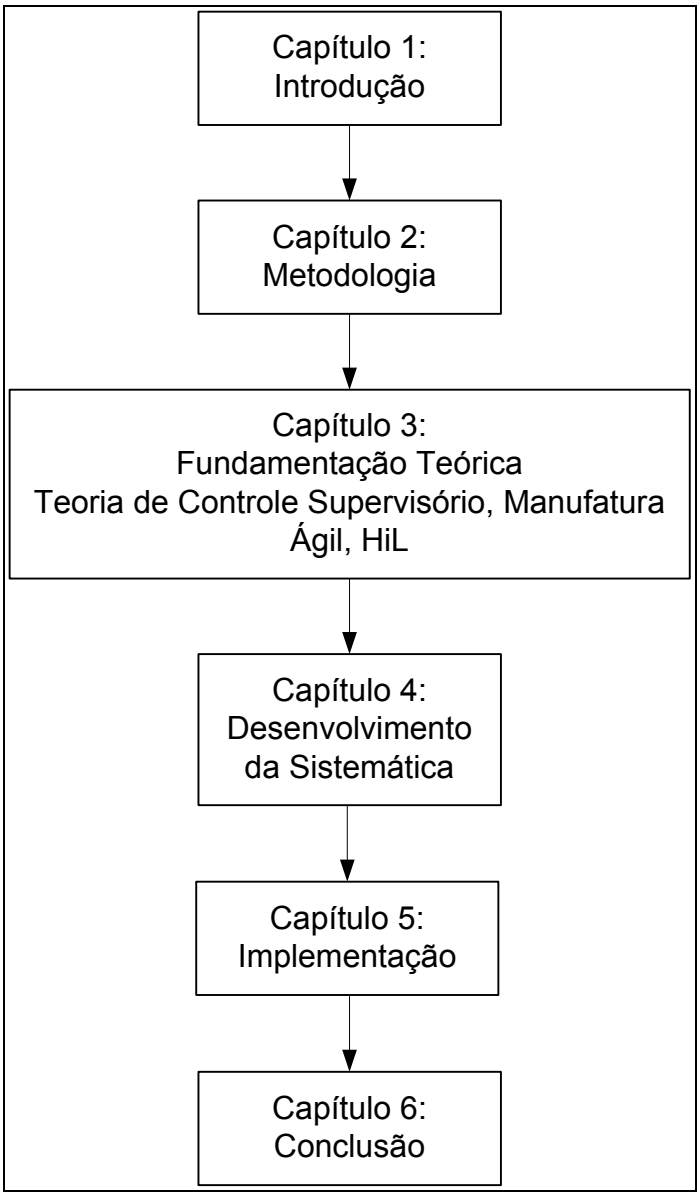


Figura 1.1 – Estruturação da dissertação

2 FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo são explicitados os conceitos teóricos fundamentais para o desenvolvimento da sistemática proposta, sendo abordados os conceitos de Teoria de Controle Supervisório que irão definir a maneira de controlar a evolução de uma planta. A simulação e a metodologia *Hardware-in-the-Loop* são implementadas para fundamentar o uso da simulação virtual inicial antes de ser iniciada a implementação em um ambiente real.

2.1 Sistemas a Eventos Discretos

Os Sistemas a Eventos Discretos (SEDs) percebem as ocorrências no mundo externo através da recepção de estímulos, denominados eventos. O "evento" é um conceito primitivo, cuja compreensão deve ser deixada à intuição, mais do que uma exata definição. Porém, não se pode deixar de enfatizar que um evento deve ser pensado como de ocorrência instantânea e como causador de uma transição no valor (discreto) do estado do sistema. São exemplos de eventos o início e o término de uma tarefa (mas não sua execução), a chegada de um cliente numa fila, a recepção de uma mensagem em um sistema de comunicação, um sinal de chegada de uma peça num processo industrial ou ainda o aperto de um botão pelo operador de uma máquina.

A ocorrência de um evento causa, em geral, uma mudança interna no sistema, a qual pode ser causada pela ocorrência de um evento interno ao próprio sistema, tal como o término de uma atividade ou o fim de uma temporização. Em qualquer caso, essas mudanças se caracterizam por serem abruptas e instantâneas: ao perceber um evento, o sistema reage imediatamente, acomodando-se em tempo nulo numa nova situação, em que permanece até que ocorra um novo

evento. Desta forma, a simples passagem do tempo não é suficiente para garantir que o sistema evolua. Ainda, a ocorrência destes eventos pode depender de fatores alheios ao sistema, de modo que este não tem, em geral, como prevê-los (CURY, 2001).

Diz-se que, entre a ocorrência de dois eventos consecutivos, o sistema permanece num determinado estado. A ocorrência de um evento causa então uma transição ou mudança de estado no sistema, de forma que sua evolução no tempo pode ser representada pela trajetória percorrida no seu espaço de estados. Considere como exemplo um elevador que se move entre o térreo (0), primeiro andar (1) e segundo andar (2) e executa somente dois tipos de movimentos: subir e descer. Supõe-se ainda que o elevador esteja inicialmente no térreo, e a seqüência de movimentos é representada na Figura 2.1 (KUMAR e GARG, 1995).

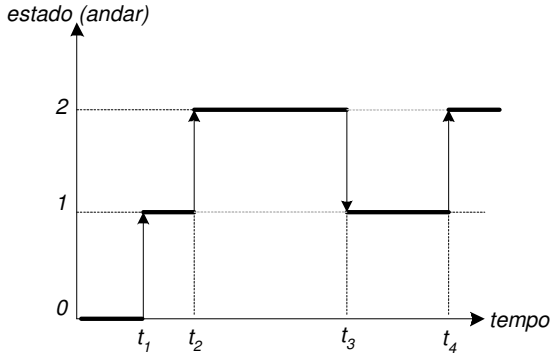


Figura 2.1 – Evolução típica de um Sistema a Eventos Discretos.

Na Figura 2.1, observa-se que o elevador tem três estados - 0,1 e 2 - e na sua trajetória ocorrem dois eventos: 'subir' e descer'. Vê-se que um mesmo evento pode ter efeitos diferentes, dependendo do estado em que ocorre. Por exemplo, se o sistema está no estado 0 e ocorre o evento subir, o próximo estado será 1; se o evento subir ocorre em 1, o próximo estado será 2. A trajetória pode continuar indefinidamente, inclusive com a ocorrência de novos eventos ou estados.

Nos sistemas tratados, assume-se que o número total de eventos diferentes que podem ocorrer é finito. Em relação ao número de estados, pode ser ilimitado no caso geral, embora a classe de sistemas com um número finito de estados seja um caso particular importante (CURY, 2001).

O espaço de estados de um SED é limitado a um conjunto enumerável e, diferentemente dos sistemas físicos, pode ter valores simbólicos em

vez de valores reais; por exemplo, uma máquina está inativa, trabalhando ou quebrada, um elevador está no térreo, primeiro ou segundo andar. Ainda, eventos (causam transições de estados) ocorrem assincronamente em instantes discretos do tempo, sendo caracterizados ou rotulados também por valores simbólicos. Desta forma, as relações entre transições de estados e eventos são irregulares e normalmente não podem ser descritas usando equações diferenciais ou de diferença, como ocorre em muitos sistemas físicos (KUMAR e GARG 1995).

Os sistemas físicos descritos por equações diferenciais são denominados sistemas dinâmicos a variáveis contínuas. Estes, em geral, mudam de estado de forma contínua, tendo o seu comportamento descrito por uma função que relaciona o estado (variável dependente) ao tempo (variável independente). Assim, tais sistemas contrastam com os SEDs, já que se caracterizam pela continuidade das variáveis de estado e cujo mecanismo de transição é dirigido pelo tempo. Ao contrário destes sistemas, os SEDs podem permanecer um tempo arbitrário em um mesmo estado, sendo que sua trajetória pode ser dada por uma sequência de eventos na forma {σ1, σ2, ...}, incluindo eventualmente os instantes de tempo em que tais eventos ocorrem {(σ1, t1), (σ2, t2), ...}. A quantidade de informação necessária depende dos objetivos da aplicação (CURY, 2001).

A Figura 2.2, extraída de CASSANDRAS e LAFORTUNE (1999) ilustra as características que distinguem os sistemas a variáveis contínuas (SVCs) dos sistemas a eventos discretos (SEDs). Para os SVCs, o espaço de estados é o conjunto dos números reais e *x(t)* poderá assumir qualquer valor deste conjunto. Também, a função *x(t)* é a solução de uma equação diferencial na forma geral $\dot{x}(t) = f(x(t), u(t), t)$, em que u(t) é a entrada. No caso de SEDs, o espaço de estados é o conjunto discreto $X = \{s_1, s_2, s_3, s_4, s_5, s_6\}$. A evolução do sistema se dá pela mudança de um estado a outro sempre que um evento ocorre. Observa-se que a ocorrência de um evento não significa necessariamente uma transição de estado, como o caso do evento e_3. Nota-se então que não existe nenhum mecanismo que especifica como eventos interagem com o tempo ou como o tempo de ocorrência pode ser determinado (CASSANDRAS e LAFORTUNE, 1999).

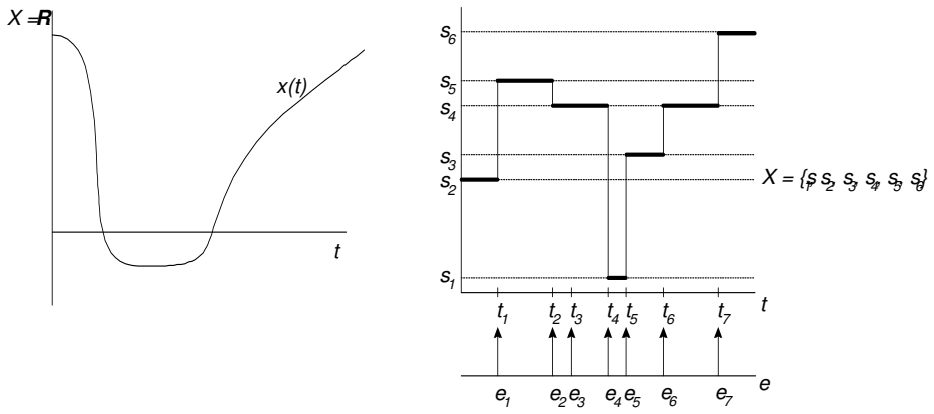


Figura 2.2 – Sistemas a Variáveis Contínuas e Sistemas a Eventos Discretos.

2.2 Controle supervisório de sistemas a eventos discretos

Esta seção apresenta a Teoria de Controle Supervisório de Sistemas a Eventos Discretos, formulada inicialmente por RAMADGE e WONHAM (1989).Esta teoria tem sido desenvolvida nas últimas décadas como uma proposta de metodologia formal de síntese de controladores ótimos para SEDs, entre os quais se inclui grande parte dos sistemas de manipulação e montagem automatizados.

Na abordagem proposta por RAMADGE e WONHAM (1989) (abordagem RW), o SED a ser controlado, ou planta na terminologia de controle tradicional, é representado por uma linguagem gerada *L* (seqüências parciais) e por uma linguagem marcada *Lm* (tarefas completadas). Conforme discutido na seção anterior, assume-se aqui que a planta *G* é modelada por um autômato. A notação *G* será então usada indistintamente para referenciar a planta ou o seu modelo em autômato.

Dessa forma, as linguagens *L(G)* e *Lm(G)* podem conter cadeias indesejáveis de eventos por violarem alguma condição que se deseja impor ao sistema. Pela junção de uma estrutura de controle (supervisor), será possível modificar a linguagem gerada pelo sistema dentro de certos limites, evitando aquelas cadeias indesejadas de eventos. A característica de controle é introduzida ao se considerar que certos eventos podem ser desabilitados por um controlador externo.

Assim, pode-se influenciar na evolução do SED pela proibição da ocorrência de eventos-chave em certos momentos.

O autômato *G* modela então o comportamento não controlado do SED, ou o comportamento em malha aberta analogamente à teoria de controle clássica. A premissa é que este comportamento não é satisfatório e deve ser modificado por uma ação de controle. A modificação deste comportamento deve ser entendida como uma restrição do comportamento a um subconjunto de *L(G)*. Para alterar o comportamento introduz-se um supervisor, denotado por *S*.

A idéia central é construir um supervisor tal que os eventos que ele desabilita num dado instante dependem do comportamento passado do SED. Refere-se a esta abordagem como controle supervisório monolítico, pois o objetivo é projetar um único controlador cuja função é habilitar e desabilitar certos eventos, conforme a seqüência de eventos observados na planta.

Dentro desta abordagem, considera-se que o supervisor *S* interage com a planta *G*, numa estrutura em malha fechada, onde *S* observa os eventos ocorridos em *G* e define que eventos, dentre os fisicamente possíveis de ocorrerem no estado atual, são permitidos de ocorrerem a seguir. Sob este aspecto, a forma de controle é dita permissiva, no sentido que eventos inibidos não podem ocorrer e os autorizados não ocorrem obrigatoriamente. O conjunto de eventos habilitados num dado instante pelo supervisor define uma entrada de controle. Esta é atualizada a cada nova ocorrência de evento observada em *G*. A Figura 2.3 a seguir ilustra o acoplamento entre a planta e o supervisor.

Estes conceitos levam à distinção do sistema a controlar (planta) e do agente de controle (supervisor), permitindo assim distinguir o comportamento fisicamente possível do sistema e as restrições ligadas a comportamentos não desejados.

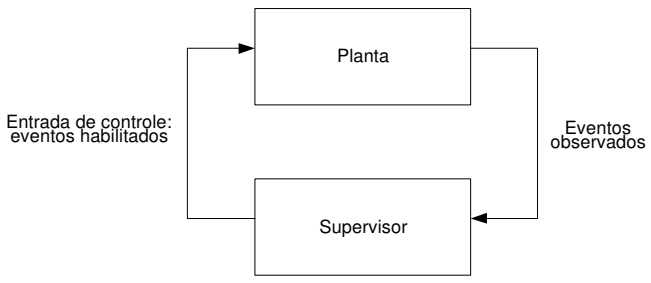


Figura 2.3– Acoplamento da planta e supervisor no modelo RW (RAMADGE e WONHAM, 1989).

Para associar a um SED ou a uma planta *G* estruturas de controle, particiona-se o alfabeto Σ em um conjunto Σc de eventos controláveis que podem ser inibidos de ocorrer, e um conjunto Σu de eventos não controláveis, sobre os quais o agente de controle não tem influência. Para que seja possível interferir no funcionamento da planta *G*, este precisa ser dotado de uma interface através da qual se possa informar quais eventos devem ser habilitados e quais devem ser inibidos. Considera-se o conjunto de eventos que se deseja habilitar como uma entrada de controle. Naturalmente, esta entrada de controle não inibe eventos não controláveis. Formalmente define-se uma estrutura de controle associada a *G* como o conjunto de entradas de controle:

$$\Gamma = \left\{ \gamma \in 2^{\Sigma} : \gamma \supseteq \Sigma_u \right\}$$

onde a condição $\gamma \supseteq \Sigma_u$ indica simplesmente que os eventos não controláveis são necessariamente habilitados.

Quando se aplica uma entrada de controle γ a uma planta, esta se comporta como se os eventos inibidos fossem momentaneamente apagados da sua estrutura de transição, afetando com isso a linguagem gerada. É este o princípio de funcionamento do mecanismo de controle adotado no modelo RW, que consiste em chavear as entradas de controle em resposta ao comportamento observado do sistema, de modo a confinar a linguagem gerada a uma dada especificação. Considera-se então que a função de transição de um autômato cujo alfabeto foi particionado em eventos controláveis e eventos não controláveis deixa de ser definida para os eventos inibidos por uma entrada de controle aplicada, enquanto esta estiver presente, sem explicitar este fato na notação.

O agente controlador é denominado supervisor. Formalmente, um supervisor *f* é um mapeamento $f: L \rightarrow \Gamma$ que especifica, para cada cadeia possível de eventos gerados $w \in L$, um conjunto de eventos habilitados (entrada de controle) $\gamma = f(w) \in \Gamma$. O SED *G* controlado por *f* é denotado por *f/G*. O comportamento do sistema sob a ação do supervisor, definido pela linguagem $L(f/G) \subseteq L(G)$ satisfaz: $i) \varepsilon \in L(f/G)$ e $w\sigma \in L(f/G)\ sse\ w \in L(f/G), w\sigma \in L(G) e \sigma \in f(w)$.

Diz-se que um supervisor f para a planta G é não bloqueante se, e somente se, $\overline{L_m(f/G)} = L(f/G)$. Isso implica que, de qualquer estado do

comportamento em malha fechada da planta, uma tarefa pode ser completada no sistema.

Pode-se representar um supervisor por um autômato *S*, definido sobre o mesmo alfabeto Σ, cujas mudanças de estado são ditadas pela ocorrência de eventos na planta *G*. A ação de controle de *S*, definida para cada estado do autômato, é desabilitar em *G* os eventos que não possam ocorrer em *S* após uma cadeia de eventos observada. O supervisor *S* pode ser interpretado como um autômato que aceita como entrada os eventos gerados por *G* e executa transições de estado de acordo com sua função de transição.

O funcionamento do sistema controlado *S/G* pode ser descrito por um SED resultante da composição síncrona de *S* e *G*, isto é, *S* // *G*. De fato, na composição síncrona *S* // *G* somente as transições permitidas tanto no sistema controlado *G*, como no supervisor *S* podem ocorrer.

O comportamento em malha fechada do sistema é então dado por:

$L(S/G) = L(S \| G)$ e $L_m(S/G) = L_m(S \| G)$.

De um modo geral, um problema de síntese de supervisores supõe que se represente o comportamento fisicamente possível do sistema e o comportamento desejado sob supervisão por linguagens, sendo o objetivo construir um supervisor para a planta de forma que o comportamento do sistema em malha fechada se limite ao comportamento desejado. Formalmente, o Problema de Controle Supervisório (PCS) é apresentado a seguir:

Dada uma planta *G*, com comportamento *(L(G), Lm(G))* e uma estrutura de controle Γ (conjunto de entradas de controle), definidos sobre o conjunto de eventos Σ; e especificações definidas por $A \subseteq E \subseteq \Sigma^*$; encontrar um supervisor não bloqueante *S* para *G* tal que

$A \subseteq Lm\ (S/G) \subseteq E$.

As especificações *A* e *E* definem limites inferior e superior para o comportamento do sistema em malha fechada .

As especificações são interpretadas da seguinte forma. A linguagem gerada *L(G)* contém palavras que não são aceitas, pois violam alguma condição que se deseja impor ao sistema. Pode ser certos estados de *G* que são indesejados e devem ser evitados, sendo estes estados causadores de bloqueio ou então fisicamente inadmissíveis, por exemplo, a colisão de um robô com um veículo auto

guiado ou a tentativa de colocar uma peça num *buffer* cheio num sistema de manufatura automatizado. Ou ainda, algumas palavras em *L(G)* podem conter prefixos que não são permitidos, que violam uma seqüência desejada de certos eventos. Assim, consideram-se sub-linguagens de *L(G)* que representam o comportamento 'legal' ou 'admissível' do sistema controlado (CASSANDRAS e LAFORTUNE, 1999).

2.3 Controle modular local

A teoria de controle supervisório proposta por Ramadge e Wonhan (1989) possui a vantagem de permitir a síntese automática de supervisores e, também, a noção de máxima linguagem controlável garante a síntese de controladores de forma minimamente restritiva. No entanto, quando um grande número de tarefas deve ser executado pelo sistema de controle, a abordagem monolítica pode ter um desempenho computacional bastante desfavorável. Isso porque a composição síncrona das especificações gera um crescimento exponencial no número de estados do modelo e, por conseguinte, na complexidade computacional do problema.

Uma forma de diminuir a complexidade da síntese de controladores é dividir a tarefa de controle em várias sub-tarefas, que são resolvidas usando a teoria de controle segundo Ramadge e Wonhan (1989), e combinar os sub-controladores resultantes de modo a solucionar o problema original. Esta concepção é chamada síntese modular e os controladores resultantes de supervisores modulares. Esta abordagem foi introduzida por Ramadge e Wonhan (1988) e é referida como a teoria de controle modular.

A síntese modular permite, assim, que problemas complexos possam ser decompostos em módulos mais simples, de forma a atribuir maior flexibilidade ao controlador resultante. Além de ser mais facilmente construído, um supervisor modular costuma ser mais facilmente modificado, atualizado e corrigido. Por exemplo, se uma sub-tarefa for mudada, só é preciso reconstruir o subcontrolador correspondente ao invés de refazer todo o sistema supervisor.

Em contrapartida, os controladores modulares têm suas ações de controle baseadas numa versão parcial do estado de funcionamento do sistema global. Por conseguinte, a síntese modular é, em geral, degradada em relação à solução monolítica, podendo em muitos casos gerar conflitos na ação de controle. A chave para garantir o não bloqueio entre controladores é a propriedade de *modularidade*. Quando esta condição é verificada, o controle modular mostra-se bastante vantajoso em relação ao monolítico em termos da implementação da estrutura de controle e da complexidade do processo de síntese. O conceito de modularidade é apresentado a seguir:

Sejam duas linguagens *L1, L2* $\subseteq \Sigma^*$. É sempre verdade que $\overline{L_1 \cap L_2} \subseteq \overline{L_1} \cap \overline{L_2}$, isto é, o prefixo de uma cadeia comum a *L1* e *L2* é também um prefixo de *L1* e *L2*. Diz-se que *L1* e *L2* são modulares (ou não conflitantes) se $\overline{L_1 \cap L_2} = \overline{L_1} \cap \overline{L_2}$. Isso quer dizer que duas linguagens são modulares se, toda vez que compartilham um prefixo, compartilham também uma palavra contendo este prefixo. Por exemplo, quaisquer linguagens prefixo fechadas são modulares entre si.

Quando a propriedade de modularidade entre as tarefas de controle é verificada, o controle modular mostra-se bastante vantajoso em relação ao monolítico em termos de implementação da estrutura de controle e da complexidade do processo de síntese. No entanto, a modelagem por autômatos pode induzir a uma explosão de estados à medida que subsistemas vão sendo agregados a ela. Apesar de cada supervisor ser concebido para uma especificação isolada, a abordagem considera que cada módulo de controle observa e controla a planta inteira. Dessa forma, o controle modular pode ser inviável para sistemas de grande porte.

Queiros e Cury (2002) propõem uma solução alternativa para a síntese de controle modular que explora, além da modularidade das especificações, a modularidade da planta. A abordagem proposta pelos autores é denominada controle modular local, onde é proposta uma arquitetura distribuída em que cada módulo de controle atua somente sobre os subsistemas atingidos. Dessa forma, o controle modular proposto por Queiros e Cury (2002) é uma abordagem adequada quando comparada ao controle monolítico e mesmo ao modular clássico, uma vez que permite uma redução da complexidade computacional tanto no processo de

síntese quanto no funcionamento da estrutura de controle. É esta abordagem que é seguida no presente trabalho.

2.4 Modelagem de sistemas compostos

Segundo Queiroz (2000), no projeto de sistemas de maior complexidade, a modelagem das diversas partes envolvidas é geralmente um passo intermediário na representação do comportamento conjunto do sistema. Isso porque a modelagem dessas partes exige menor esforço computacional, menos memória e costuma ser mais compreensível para o projetista.

Tais sistemas são normalmente modelados pela composição de subsistemas de menor porte, podendo estes subsistemas ser assíncronos entre si. Dessa forma, de acordo com o nível de composição das sub-plantas originalmente modeladas, diferentes representações para o sistema podem ser formuladas. Na abordagem de controle modular local, são utilizadas duas representações para o sistema: a representação por sistemas compostos (RSC) e a representação por sistemas produto (RSP) (Ramadage e Wonhan, 1989). A definição destas duas representações é dada a seguir.

Representação por Sistema Composto (RSC)

É qualquer modelagem da planta global $G=(\Sigma,Q,\delta,q_0,Q_m)$ obtida pela combinação de sub-plantas $G'=(\Sigma'_i,Q'_i,\delta'_i,q'_0,Q'_m)$, $i\in N'=\{1,...,n'\}$. Assim, tem-se $G=\|_{i=1}^{n'} G'_i$, com alfabeto de eventos $\Sigma=\bigcup_{i=1}^{n'}\Sigma'_i$.

Representação por Sistema Produto (RSP)

Um Sistema Produto é um sistema que pode ser modelado pela composição de subsistemas completamente assíncronos entre si (Ramadge e Wonhan, 1989; Ramadge, 1989). Denomina-se Representação por Sistema Produto (RSP) qualquer RSC cujas subplantas não tenham eventos síncronos em comum.

Considerando que cada subsistema de uma RSP representa uma parte isolada de um sistema em malha aberta, pode-se afirmar que esta modelagem representa a estrutura descentralizada natural de operações concorrentes para sistema de maior porte. Pode-se obter a mais refinada Representação por Sistema Produto equivalente a uma RSC qualquer, pela composição dos geradores síncronos deste último. Para isso, faz-se a composição dos subsistemas síncronos originais (que têm eventos em comum), criando-se o maior número possível de subsistemas assíncronos, cada qual com o mínimo de estados (Queiroz, 2000).

As definições apresentadas a seguir fundamentam o controle modular local de sistemas compostos.

Considere a RSP de um sistema G formada por subsistemas $G_i = (\Sigma_i, Q_i, \delta_i, q_{0i}, Q_{mi})$, $i \in N = \{1,...,n\}$. Para $j = 1,...,m$, sejam especificações genéricas modeladas por E_{xj} definidas respectivamente em $\Sigma_{xj} \subseteq \Sigma$. Para $j = 1,...,m$, a planta local $G_{Xj} = (\Sigma_{Xj}, Q_{Xj}, \delta_{Xj}, q_{0Xj}, Q_{mXj})$ associada à especificação Σ_{xj} é definida por $G_{Xj} = \underset{i \in N_{Xj}}{\|} G_i$, com $N_{Xj} = \{k \in N \,|\, \Sigma_K \cap \Sigma_{xj} \neq \varnothing\}$. Assim, a planta local G_{Xj} é composta apenas pelos subsistemas da modelagem original que estão diretamente (e indiretamente) restringidos por E_{xj}.

Exemplo: Seja uma RSC de um sistema formada pelo conjunto de subplantas $\{G'_i = (\Sigma'_i, Q'_i, \delta'_i, q'_{0i}, Q'_{mi}), i = 1,...,5\}$. Sejam duas especificações genéricas $E_a \subseteq \Sigma_a^*$ e $E_b \subseteq \Sigma_b^*$. A relação entre os conjuntos de eventos é ilustrada na Figura 2.4.

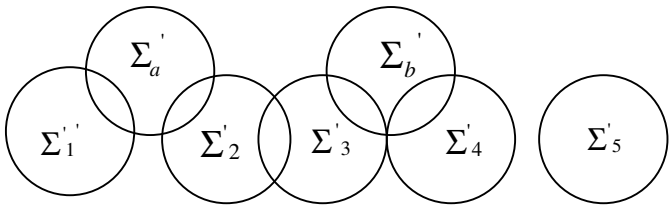


Figura 2.4- Alfabeto de eventos de um sistema composto.

Esse sistema tem uma RSP mais refinada dada pelo conjunto de plantas assíncronas $\{G_i = (\Sigma_i, Q_i, \delta_i, q_{0i}, Q_{mi}), i = 1,...,4\}$, onde $G_1 = G'_1$, $G_2 = G'_2 \| G'_3$, $G_3 = G'_4$ e $G_4 = G'_5$. Assim, as plantas locais são dadas por $G_A = G_1 \| G_2$ e $G_B = G_2 \| G_3$.

Deste modo, podem ser calculadas as especificações locais $E_A = E_a \| L_m(G_A)$ e $E_B = E_b \| L_m(G_B)$.

2.5 Implementação de supervisores locais

Os supervisores resultantes do processo de síntese apresentados na seção anterior são descritos como máquinas de estados finitos em que, para cada estado ativo, um conjunto de eventos controláveis deve ser desabilitado. Desta forma, a implementação do programa de controle consiste basicamente em fazer o CLP se comportar como um jogador de autômatos.

Segundo a proposta apresentada por Queiroz et al. (2001), a estrutura de controle é desenvolvida em três níveis estruturais: o nível dos Supervisores Modulares que desabilitam eventos da planta de acordo com as mudanças de estado da mesma; o nível de Sistema Produto que, seguindo os modelos supervisionados das plantas, é responsável por comandar o início das seqüências de operação; e o nível das Seqüências Operacionais que, observando os sinais de entrada do CLP e ajustando os sinais de saída, executa os ciclos de funcionamento de cada dispositivo. Essa estrutura de controle é ilustrada na Figura 2.5.

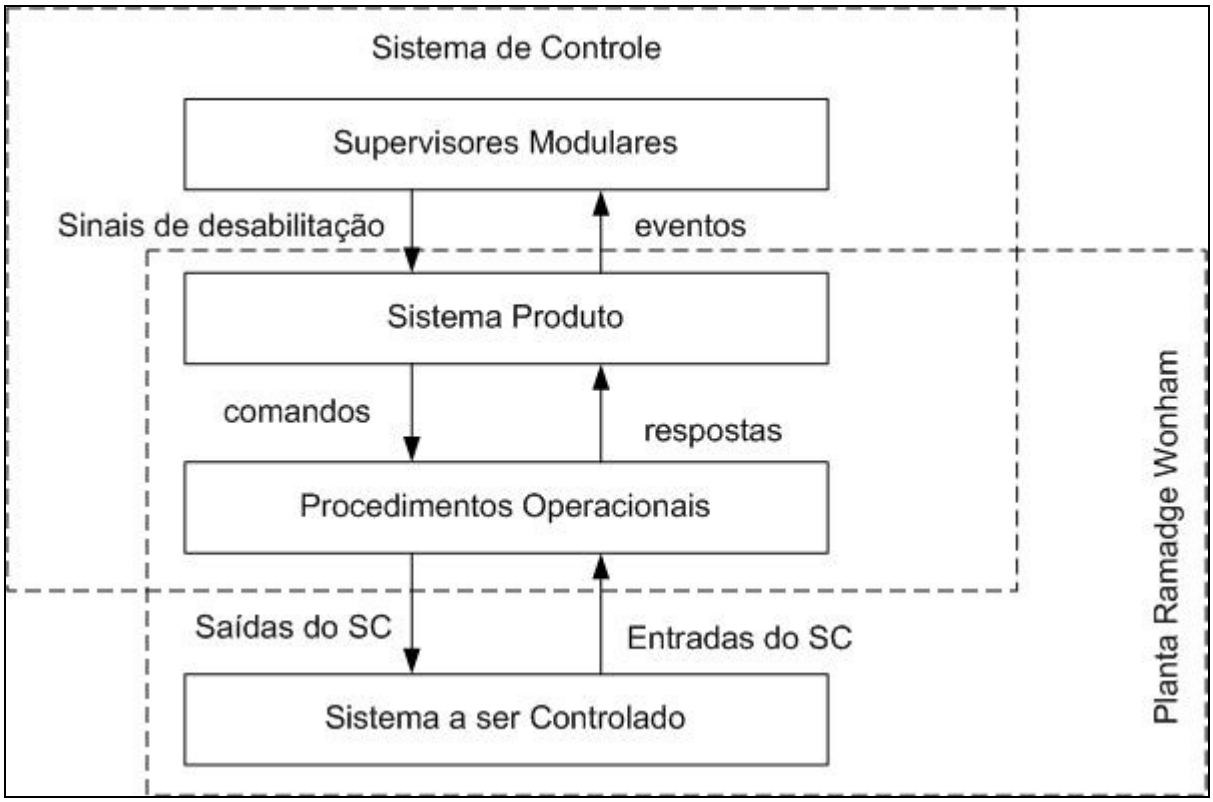


Figura 2.5– Estrutura básica do sistema de controle (Queiroz et al., 2001).

Os subsistemas G_i são então representados como máquinas de estados assíncronas. As transições controláveis são automaticamente disparadas quando não são desabilitadas pelos supervisores. As transições não controláveis são disparadas por algum sinal da planta. Cada transição também sinaliza uma mudança de estado do supervisor, atualizando-o continuamente. Desta forma, evita-se que duas transições seguidas ocorram no sistema produto sem que os supervisores tenham sido atualizados.

2.6 Estrutura de Controle proposta por Vieira (2004)

A princípio, a teoria de controle supervisório (Ramadge and Wonham, 1989) sugere que uma estrutura de controle consiste apenas da implementação dos supervisores (Figura 2.6-a). Os eventos seriam gerados pelo sistema físico, determinando o conjunto de estados ativos dos supervisores. Em

função deste conjunto seriam definidos e enviados sinais de desabilitação de eventos controláveis ao sistema. Entretanto duas grandes hipóteses determinam que esta estrutura simplificada não seja eficaz: a suposição de geração espontânea de eventos e a abstração realizada durante a modelagem do sistema físico.

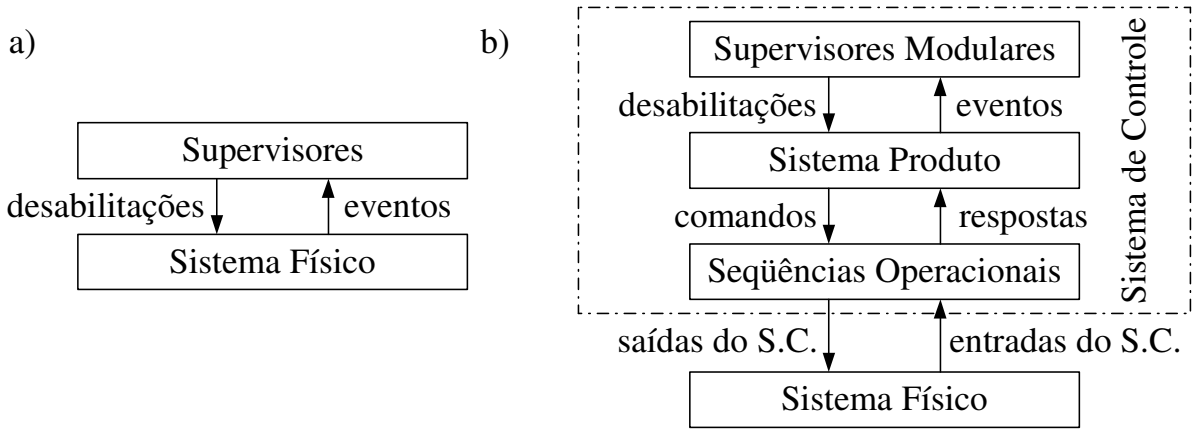


Figura 2.6 – Estrutura de controle

A estrutura de controle proposta em (Queiroz e Cury, 2002) resolve estas questões através da implementação de uma interface entre os supervisores e o sistema físico, o que resulta na estrutura de controle conforme apresentado na Figura 2.6-b. O nível denominado “Supervisores Modulares” (SM) corresponde ao conjunto de autômatos ($S=\{Sr_i|i=1,...,m\}$) e respectivas funções ψr_i que representam os supervisores reduzidos obtidos conforme o procedimento descrito na seção anterior. A interface é constituída pelo nível “Sistema Produto” (SP) e o nível “Seqüências Operacionais” (SO).

Para implementar esta interface deve ser realizado inicialmente um mapeamento relacionando: *i)* a cada evento controlável de Σ_{cj} uma *desabilitação*; *ii)* a cada ocorrência de evento controlável em G_j um *comando*; *iii)* a cada ocorrência em G_j de evento não controlável uma *resposta*. Este mapeamento define, respectivamente, três funções bijetoras para cada módulo $G_j : fd_j : \Sigma_{cj} \rightarrow \Phi_j$; $fc_j : Q_j x \Sigma_{cj} \rightarrow \Delta_j$; $fr_j : Q_j x \Sigma_{uj} \rightarrow \Pi_{rj}$; onde: Φ_j – alfabeto de *desabilitações*; Δ_j – alfabeto de *comandos*; Π_{rj} – alfabeto de *respostas* e Σ_{cj}, Σ_{uj}, Q_j conforme definidos na Seção 2. Uma quarta função deve ser definida relacionando a cada *desabilitação* uma *negação de desabilitação* $fnd_j : \Phi_j \rightarrow \Pi_{ndj}$, onde Π_{ndj} – alfabeto de *negação de desabilitações*.

As estruturas de implementação do nível SP podem ser representadas pelo conjunto de autômatos G={g_j|j=1,...,p}, o qual é obtido do conjunto {G_j|j=1,...,p}. A cada autômato G_j há relacionado um autômato g_j na forma de máquina de Mealy: $g_j = (\Pi_j, 2^{\Gamma_j}, Q_j, q_{0j}, \xi_j, \omega_j)$ onde: Π_j é o alfabeto de entrada, com $\Pi_j = \Pi_{ndj} \cup \Pi_{rj}$; 2^{Γ_j} é o alfabeto de saída, com $\Gamma_j = \Sigma_j \cup \Delta_j$; ξ_j é a função de transição de estados na forma $\xi_j : Q_j x \Pi_j \to Q_j$; e ω_j é a função de saída na forma $\omega_j : Q_j x \Pi_j \to 2^{\Gamma_j}$. Nestes autômatos Q_j e q_{0j} são idênticos ao conjunto de estados e ao estado inicial do autômato G_j correspondente.

A função de transição de estados ξ_j do autômato g_j é definida por:

$$\xi_j(q, \neg\alpha d) = q' \quad \text{sempre que} \quad \delta_j(q, \alpha) = q'$$
$$\text{com}: \neg\alpha d = \text{fnd}_j(\text{fd}_j(\alpha));$$
$$\xi_j(q, \text{rpl}\beta) = q' \quad \text{sempre que} \quad \delta_j(q, \beta) = q'$$
$$\text{com}: \text{rpl}\beta = \text{fr}_j(q, \beta).$$

Ou seja: cada ocorrência de evento controlável em δ_j é substituída pela *negação da desabilitação* correspondente; cada ocorrência de evento não controlável é substituída pela *resposta* correspondente.

A função de saída ω_j é definida por:

$$\omega(q, \neg\alpha d) = \{\alpha, \text{cmd}\alpha\}$$
$$\text{com}: \neg\alpha d = \text{fnd}_j(\text{fd}_j(\alpha)); \ \text{cmd}\alpha = \text{fc}_j(q, \alpha);$$
$$\omega(q, \text{rpl}\beta) = \{\beta\}$$
$$\text{com}: \text{rpl}\beta = \text{fr}_j(q, \beta).$$

Ou seja: a cada transição de estado de g_j relacionada a uma *negação de desabilitação* serão gerados o *evento* e o *comando* correspondentes; a cada transição de estado de g_j relacionada a uma *resposta* será gerado o *evento* correspondente.

Durante a etapa de modelagem do sistema físico é recomendado realizar uma abstração dos detalhes de implementação tecnológica das diversas atividades e funções realizadas pelo sistema físico. Este detalhamento é realizado no nível SO. Assume-se, sem perda de generalidade, que a cada subsistema físico há uma seqüência operacional correspondente que realiza este detalhamento. O nível SO corresponde então a este conjunto de seqüências operacionais (O={o_j|j=1,...,p}).

Uma possível forma de realizar a descrição de uma seqüência operacional é através de um *sequential function chart* (SFC) (ISO/IEC, 1993). A elaboração de cada SFC deve ser realizada de forma a garantir a coerência com o

correspondente autômato G_j. Neste sentido, deve haver uma relação direta entre a linguagem do autômato G_j e as seqüencias de *comandos* e *respostas* que são obtidas através das possíveis seqüências de evolução do SFC o_j correspondente. Além disto: *i)* cada *comando* estará relacionado a uma, e somente uma, condição de transição do SFC; *ii)* cada *resposta* deverá ser ativada com retenção em um, e somente um, passo do SFC que corresponde à ocorrência do evento não controlável representado em G_j; *iii)* todas as *respostas* deverão ser desativadas no passo sucessor de cada transição do SFC relacionada a um *comando*.

Seja a estrutura de controle conforme apresentado na Figura 2.6 -b. Durante a transição de estado do autômato que representa um determinado módulo do sistema produto é gerado um *evento*, se esta transição for associada a uma negação de desabilitação de evento controlável também será gerado o *comando* correspondente. Conforme representado nesta figura, o *evento* é encaminhado para o nível SM, determinando a transição de estado dos autômatos que representam os supervisores reduzidos. O *comando* é encaminhado para o nível SO, determinando a evolução do SFC utilizado para descrever a correspondente seqüência operacional.

Em função do estado ativo de cada autômato Sr_i a correspondente função ψr_i estabelece a desabilitação de um subconjunto de eventos controláveis. A *desabilitação* de evento controlável é ativada se for estabelecida por pelo menos um supervisor $(Sr_i,\psi r_i)$.

A evolução do SFC utilizado para representar uma dada seqüência operacional é determinada conforme a execução de atividades ou tarefas no Sistema Físico. Ao longo da sua evolução este SFC atingirá um determinado passo no qual será gerada uma *resposta.* Esta *resposta* será encaminhada para o nível SP permitindo a transição de estado do correspondente módulo do sistema produto.

De forma a garantir a correta operação da estrutura de controle duas condições devem ser obedecidas:

i) a evolução do sistema produto através de um evento controlável só pode ocorrer se todos os supervisores que podem exercer ação de desabilitação sobre este evento estiverem atualizados e se nenhum destes supervisores estiver desabilitando este evento (garante a ação conjunta dos supervisores);

ii) em cada um dos supervisores, só é permitida a evolução de estado através de um único evento por ciclo de varredura do CLP (garante a correta

evolução dos supervisores para o atual modelo de implementação dos mesmos e a conformidade com a teoria de controle supervisório, a qual não prevê a ocorrência simultânea de eventos).

É possível que, simultaneamente, hajam diversos eventos controláveis habilitados, ou diversas respostas ativas, ou ainda, um ou mais eventos controláveis habilitados e uma ou mais respostas ativas. Assim, de forma a garantir a satisfação das condições *(i)* e *(ii)* listadas acima, sempre que houver a geração de um evento através da evolução de um determinado módulo g_i deve haver a desabilitação de evolução de todos os módulos do sistema produto que definem uma planta local com o referido módulo até que ocorra a atualização de estado dos supervisores associados a estas plantas locais. Isto implica que a evolução de um módulo do sistema produto pode ser adiada durante um ou mais ciclos de varredura do CLP.

Em termos de implementação do programa, sempre que houver a evolução de um módulo do sistema produto será ativada uma variável que sinaliza a evolução do módulo em questão. A desabilitação de evolução de cada módulo do sistema produto é determinada se algum módulo que compõe uma planta local com o módulo em questão tem sua evolução sinalizada.

A consideração da possibilidade de que a evolução de um módulo do sistema produto através de uma transição associada a uma resposta seja adiada por um ou mais ciclos de varredura do CLP justifica a necessidade de ativação com retenção das *respostas* geradas no nível SO e sua manutenção no estado ativo até que seja confirmado que a correspondente transição de estado tenha sido executada. O mecanismo atualmente adotado para estabelecer esta confirmação é a geração de um comando associado ao módulo em questão, porém o mesmo implica na seguinte restrição: *um caminho fechado (ciclo) no autômato G_j não pode ser constituído exclusivamente de eventos não-controláveis.* Caso esta restrição seja violada é possível que, após ativada, uma determinada resposta permaneça ativa indefinidamente. Com isto, a transição de estado do autômato g_j, associada a esta resposta, pode ser disparada indevidamente caso o estado que a precede seja novamente ativado.

2.7 Ciclo de Desenvolvimento de sistemas automatizados e integrados

Busetti e Santos (2006) propõem a utilização de um ciclo de desenvolvimento para processos reconfiguráveis composto por três etapas, a fim de garantir uma manufatura ágil e de forma a manter uma contínua melhoria nos processos e modelos. Esse ciclo de desenvolvimento foi elaborado com a intenção de prover a possibilidade de reconfiguração de sistemas baseados na estrutura de controle supervisório, atendendo os projetos de manufatura que necessitam de alta eficiência e confiabilidade.

Essas três etapas são divididas em modelagem, síntese e implementação, observado na Figura 2.7.

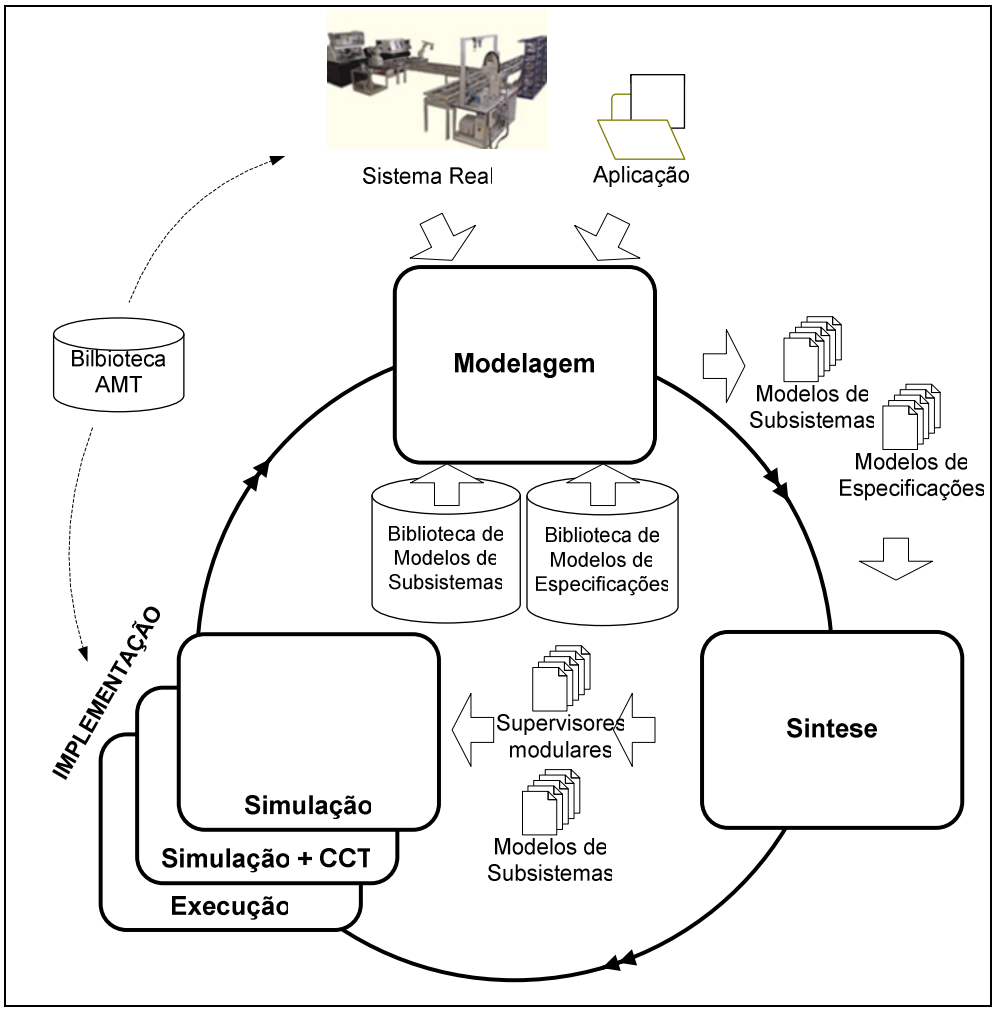


Figura 2.7 – Ciclo de desenvolvimento para um FMS

A etapa da modelagem é aquela em que os subsistemas são representados através de autômatos segundo a TCS, inicialmente desenvolvida por Ramadge e Wonham, (1989). Esses subsistemas são selecionados com base nos subsistemas físicos reais. Nessa etapa também são modeladas as especificações da planta, e assim compor os modelos de especificações. Tanto os subsistemas e especificações modeladas são armazenados em um banco de dados, biblioteca de modelos de subsistemas e especificações e reaproveitados em eventuais projetos subseqüentes. Desse modo, alguns subsistemas podem deixar de ser usados momentaneamente e, futuramente, utilizados conforme demanda.

Os modelos são feitos por meio da representação de sistema-produto (RSP), usando a composição dos subsistemas síncronos originais que apresentam eventos em comum para que haja a criação de um número maior de subsistemas assíncronos com o menor número de estados possíveis. Dessa forma, o controlador apresenta um esforço computacional menor

Na etapa de síntese são obtidos os supervisores modulares locais que atendem as especificações identificadas na etapa anterior. Esses supervisores são obtidos com base na Teoria de Controle Supervisório Modular Local Por sua vez, a etapa de implementação é dividida em três fases. Na fase de simulação, todos os subsistemas já modelados e sintetizados nas etapas anteriores são simulados em ambiente totalmente virtual. A fase de simulação +CCT é aquela em que parte dos sistemas simulados são migrados para o ambiente real e controlados por meio da aplicação de uma metodologia de simulação em tempo real. Por fim, a fase de execução é aquela em que todos os subsistemas são implementados no ambiente real.

A partir dessa última etapa, os dados gerados pela implementação são analisados e adicionados às bibliotecas de modelos, para que estas possam ser aproveitadas já com as melhorias verificadas durante o ciclo de desenvolvimento.

2.8 Simulação

A simulação é uma das técnicas que solucionam o problema por meio de um modelo que descreve o comportamento do sistema, utilizando geralmente processos computacionais. A simulação é uma técnica amplamente utilizada em vários ramos da engenharia e de outras áreas do conhecimento para avaliar o desempenho de um sistema, pois permite avaliar uma grande variedade de serviços sob diversas condições. Ela facilita a alteração do sistema sem a necessidade de alterações físicas e facilita a redução do tempo de experimentos por meio da simulação com passo de tempo avançado, podendo reproduzir em minutos o que, num sistema físico, levarias dias ou anos.

A simulação é um método eficaz, pois em muitos casos, o custo da simulação é mais barato do que o das máquinas reais e garante maior segurança, já que eventuais falhas ou problemas permanecem no mundo virtual e não danificam nem causam problemas no *hardware* real, além de serem mais rápidas que as simulações diretas em equipamento, pois não dependem do tempo real como base para evoluir na sua análise, provendo um método investigativo, no qual as soluções são previamente analisadas antes de serem colocadas em prática

Lin Li (2006) explica que o desenvolvimento da tecnologia computacional e uso mais extensivo de sistemas de produção virtual estão encorajando os estudos e as pesquisas em simulação e engenharia virtual. Para este autor, porém, no momento, poucas pesquisas são desenvolvidas, o que sugere a necessidade de se desenvolver um ambiente de simulação para VPSs abordando o controle supervisório. Em seu trabalho, Lin Li (2006) propõe esse ambiente, de forma a permitir um simulação que garantam o uso da TCS, utilizando uma estrutura de controle, combinando distribuição hierárquica e distribuída e usando linguagem autômata para a especificação dos modelos.

Para K. Iwata et al (1995), os sistemas de manufatura virtuais são modelos de computador integrados, que representam a estrutura precisa e inteira de sistemas industriais e simulam seu comportamento físico e lógico em operação. Segundo este autor, os sistemas de manufatura virtuais são agregados de vários

hardwares de computador e *softwares* necessários para se desenvolver uma arquitetura de sistema que defina componentes funcionais exigidos para modelar e efetuar uma simulação, composto por interfaces entre seus componentes e interações entre sistemas reais e virtuais.

Investir em novas tecnologias pode ser caro e arriscado. Desta forma, é fundamental para a manutenção das indústrias o uso de recursos que garantam a eficácia e a viabilidade econômica destas novas tecnologias com um baixo custo. Isso deve ocorrer, principalmente, com pequeno ou nenhum risco, antes dos investimentos serem feitos, economizando tempo e dinheiro.

Para Souza et al (2002), um ambiente de Manufatura Ágil é proposta como um novo ambiente que aborda todo o processo de desenvolvimento, simulação e fabricação do produto, possibilitando a análise das estruturas virtuais, através de simulação, antes mesmo de implementá-las no mundo real.

Souza et al (2002) citam também que a Manufatura Virtual pode ser usada em uma grande variedade de contextos de sistemas de manufatura, e pode ser definida como a modelagem desses sistemas e de componentes com o uso efetivo de computadores, de dispositivos audiovisuais e sensores para simular ou projetar alternativas para um ambiente de manufatura visando, principalmente, prever problemas potenciais e ineficiência na funcionalidade e manufaturabilidade do produto antes que a manufatura real ocorra.

A simulação visa representar a realidade através de modelos, que são representados matematicamente por equações matemáticas ou através de programas com interfaces gráficas e recursos técnicos avanças.

A simulação pode ser implantada por um *software* que representa o comportamento de um sistema conforme as definições do cenário no qual está contido. O modelo simulado é uma representação de um sistema real com fidelidade suficiente nos itens avaliados, a fim de garantir a obtenção de um resultado satisfatório para um problema.

Após essa construção do modelo no *software* é possível a existência de erros, pois esse modelo geralmente é implementado por uma pessoa, que é sujeita cometer enganos.

Para Harrell et al (2000) é importante a verificação e validação do modelo. Para eles, devem-se validar o modelo procurando dois tipos de erros, os de

sintaxe e os de semântica. Para isso, Harrel et al (2000) sugerem que algumas técnicas de validação de erros sejam adotadas:

1) Revisar a codificação do modelo,
2) Verificar se as respostas de saídas são coerentes,
3) Verificar se as animações são coerentes com o esperado,
4) Usar os recursos de detecção de erros do próprio *software*.

Segundo os autores a validação é o processo que irá determinar a relação de proximidade entre o modelo e a realidade que o mesmo representa. Desta forma eles sugerem também a validação da proximidade do modelo com a realidade usando algumas técnicas comuns como:

1) Observação da animação;
2) Comparação com o sistema atual;
3) Comparação com outros modelos já validados;
4) Teste de degeneração e condições extremas do sistema;
5) Validação por aparência, em que pessoas que dominam o conhecimento do sistema são convidadas a opinar sobre a aparência final do resultado (normalmente usado em modelos lógicos conceituais);
6) Teste com dados históricos do sistema real;
7) Análise da sensibilidade de resposta a alterações de entradas e comparadas com o sistema real;
8) Condução de *turing tests*.

2.9 Hardware-in-the-Loop

Hardware-in-the-Loop é uma técnica de simulação cada vez mais usada no desenvolvimento e teste de sistemas real-time complexos. Para Plumper (2005), a simulação HiL pode ser considerada uma simulação capaz de exercer controle, em tempo real, sobre uma planta física que está sendo simulada. Para Misselhorn et al (2006), a finalidade de uso da simulação HiL é prover uma plataforma eficaz para desenvolver e testar sistemas de real-time. A complexidade

da planta sob o controle é incluída no teste e em seu desenvolvimento, em uma representação matemática de todos os estados do sistema dinâmico, que são as modelagens da planta simulada.

Em Loures (1999), observa-se a utilização de *Hardware in the Loop* em sua prática experimental. Segundo o autor, a simulação HiL é amplamente utilizada como instrumento de apoio a elaboração de sistemas de controle, encontrada também em sistemas mecatrônicos, por causa da complexidade das plantas.

Conforme visto em Busseti (1998), uma planta é divida em subsistemas que são modelados computacionalmente. Durante a evolução do projeto, esses subsistemas computacionais são removidos e substituídos por subsistemas reais, conforme Figura 2.8.

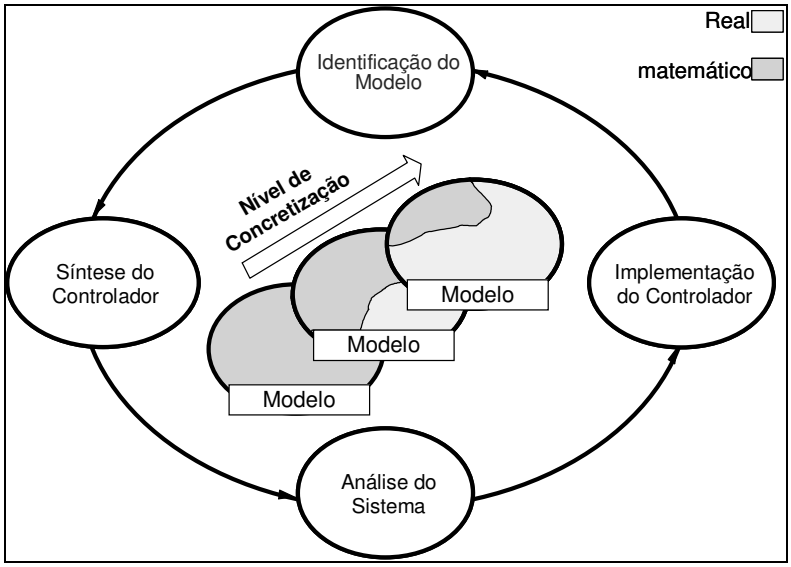


Figura 2.8 – Ciclo de desenvolvimento para simulação com HiL

É possível simular sistemas de manufatura antes de sua implementação em tempo real, trocando subsistemas por outros menores. Para Kocijan e Karba (1995), a simulação de um sistema em subsistemas menores é possível, mesmo que parte do sistema ainda não esteja completamente implantada.

Uma simulação HiL deve incluir também as simulações dos sensores e dos atuadores, agindo como relação entre a planta e o sistema testado. Os dados dos sensores são controlados pela planta simulada e lidos pelo sistema testado; da mesma forma, o sistema testado devolve os dados para a planta

simulada. As mudanças dos sinais de controle resultam em mudanças dos estados gerais da planta simulada e do sistema.

Em muitos casos, a maneira mais eficaz de se desenvolver um sistema é conectá-lo diretamente à planta física depois de simulado. Em outros casos, a simulação com HiL é mais eficiente em termos de custo, tempo de implementação e segurança. Na maioria das vezes, não há tempo hábil para montar um protótipo e, de fato, os projetos atuais necessitam de menos tempo de desenvolvimento. A simulação HiL anda em paralelo com o desenvolvimento da planta, suprindo essas necessidades.

Para Bullock et al (2002), existem três aspectos principais em uma estrutura que simula HiL. O controlador de interface, que faz a ligação entre o sistema de controle e o computador no qual esta implementada a simulação; a interface de *software*, que faz o tratamento dos sinais físicos provenientes do controlador de interface para o *software* de simulação, e a simulação microscópica, na qual os sinais provenientes do controlador de interface são usados apenas para a simulação do sistema, não exercendo nenhum efeito sobre a ação de controle.

Também para Bullock et al (2002), é necessário assegurar a sincronização entre o controlador externo e a simulação, ou seja, a simulação deve ocorrer em passos uniformes, executados em tempos menores que o tempo real, a fim de que o próximo passo possa ocorrer neste tempo.

2.10 Considerações Finais

Neste capítulo foi feita uma revisão bibliográfica sobre os principais aspectos da Teoria de Controle Supervisório, técnicas de simulação de ambientes de manufatura, a metodologia de *Hardware-in-the-Loop* e o sobre Ciclo de Desenvolvimento proposto por Busetti e Portela (2006). Dentre os aspectos principais da Teoria de Controle Supervisório, foi enfatizado que na implementação da Arquitetura de Controle Supervisório, o uso Supervisores Modulares Locais mostrou-se como a melhor evolução para a implementação em um ambiente de manufatura. Como complemento ao Ciclo de Desenvolvimento, em específico na

etapa de implementação, estruturaram-se suas fases e o uso da simulação com conceitos de manufatura, utilizando a abordagem de *Hardware-in-the-Loop*, de forma a dar forma à execução desta etapa.

Com este conjunto de fundamentos, constata-se que há viabilidade de implementação de um ambiente de simulação que auxilia a implementação da Arquitetura de Controle Supervisório no ambiente de manufatura.

3 DESENVOLVIMENTO DE UMA SISTEMÁTICA DE SIMULAÇÃO

Neste capítulo é apresentado o desenvolvimento de uma sistemática para a criação de um ambiente de simulação da Arquitetura de controle Supervisório para atuar com o ciclo de desenvolvimento proposto por Busseti & Portela (2006). Neste ambiente são simulados os níveis de sistema físico e procedimentos operacionais, também é inserido na simulação um novo nível, denominado nível de interface, quer prove conectividade entre o ambiente de simulação e o ambiente dinâmico, em que estão sendo executados os demais níveis. Desta forma é simulado o modelo de controle supervisório, garantindo a flexibilidade da configuração da planta e o seqüenciamento correto dos eventos, garantindo a correta continuidade na simulação.

3.1 Ambiente de implementação

Para Busseti & Portela (2006), o ciclo de desenvolvimento de produto, em sua etapa de implementação, é composto por três fases: simulação, simulação+CCT e Execução, conforme apresentado anteriormente.

A sistemática proposta detalha os passos que devem ser seguidos para a implantação de uma simulação capaz de validar os sistemas de manufatura baseados na teoria de controle supervisório a eventos discretos, usando as etapas deste ciclo de desenvolvimento.

Observa-se na Figura 3.1 a evolução da etapa de implementação do clico de desenvolvimento conforme proposto por Diogo el al (2008). Utilizando a arquitetura do sistema de controle proposta por Queiroz e Cury (2002), e compondo a etapa de implementação do ciclo de desenvolvimento de produto, obtém-se a evolução da simulação que é finalizada com a execução completa do sistema de

controle em ambiente real. Em Diogo el al (2008) foi proposto a criação de um ambiente dinâmico que irá efetuar o controle das comunicações entre os ambientes reais e virtuais, sendo ele também responsável por armazenar os modelos dos subsistemas, em uma biblioteca, para usos futuros.

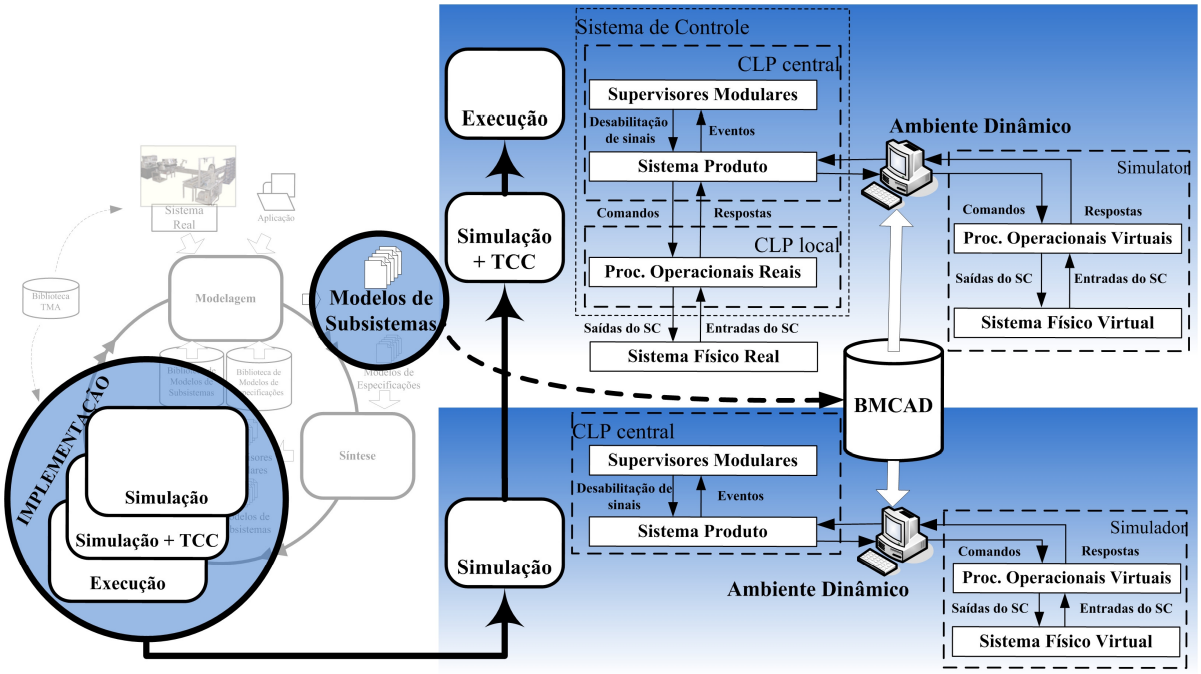


Figura 3.1 – Etapa de implementação e o Sistema de Controle – Diogo et al (2008)

Inicialmente, na fase de simulação, os Supervisores Modulares (SM) e o sistema-produto (SP) estão implementados fisicamente em um CLP, e somente os Procedimentos Operacionais (OP) e os Sistemas Físicos (SF) são simulados. Assim, todos os Procedimentos Operacionais podem ser simulados, testados e corrigidos antes de sua implementação no ambiente real.

A partir deste ponto, os procedimentos operacionais são migrados gradativamente para o ambiente real até que todos estejam completamente implementados nele e que o sistema físico real possa ser implantado por inteiro.

Observa-se na Figura 3.1 que nesta sistemática é inserido um ambiente dinâmico, responsável por fazer a interligação entre os sistemas reais e os sistemas virtuais simulados. Esse ambiente dinâmico é abordado por Diogo et al, (2008).

Para fazer que a implementação em três estágios seja possível, uma plataforma computacional deve ser concebida de forma a estabelecer uma comunicação entre o CLP e o simulador. Esta plataforma é o AD proposto por Diogo

et al (2008). Este ambiente consiste de uma plataforma computacional que realiza várias operações: *i)* Ele envia para o simulador os comandos ($Cmd\alpha_i$) gerados no CLP central; *ii)* Ele recebe as notificações de recebimento ($Ackcmd\alpha_i$) do simulador. Isto é feito para desligar o comando correspondente no CLP central; *iii)* Ele recebe as respostas ($Rps\beta_i$) do Simulador e envia para o CLP central a fim de gerar o evento não-controlável correspondente; *iv)* Ele envia para o Simulador as notificações de reconhecimentos ($AckRps\beta_i$) geradas pelo CLP central. Isto é feito para desligar no simulador a resposta correspondente; *v)* Ele tem uma interface para alternar entre o modelo de simulação e o ambiente real. Contudo, a simulação continua sendo executado em paralelo com o subsistema real. Então, o AD inverte as respostas e as notificações de recebimento quando um subsistema real está conectado. Isto é feito para impedir a geração de respostas pelo Simulador. Neste caso, o simulador recebe as repostas correspondentes do PO e gera a notificação de reconhecimento ao CLP local. Desta forma é possível adquirir dados reais do sistema de manufatura.

O AD foi desenvolvido em um *software* SCADA chamado Elipse E3 (Elipse *Software*, 2007). Essa plataforma foi escolhida devido ao fato de que inclui programas que permitem a comunicação com os periféricos (*drivers*), e que permitem a integração com outros sistemas de bancos de dados (BD), e principalmente, devido a possibilidade de gerar biblioteca de modelos. Particularmente, este recurso nos permite armazenar modelos de subsistemas (e seus respectivos sinais, de acordo com o modelo de implementação de três níveis).

A metodologia apresentada por Diogo et al (2008) faz uso de uma Biblioteca de Modelos de Comunicação do Ambiente Dinâmico (BMCAD), observado na Figura 3.1. Esta biblioteca armazena os modelos propostos no ciclo de desenvolvimento na etapa de modelagem. Ao armazená-lo, os projetistas podem usar modelos já desenvolvidos.

De acordo com o ciclo de desenvolvimento na Figura 3.1, a primeira etapa corresponde à execução da simulação da planta em estudo. Conforme descrito no capítulo dois, a simulação ocorre para validar as restrições do SM. Para tornar esta abordagem possível, o AD está conectado ao simulador para que troquem sinais entre si. Cada modelo do BMCAD é composto por um par de canais de comunicação. Neste caso, o objetivo do AD é permitir à simulação que interaja com a troca de eventos enviada pelo CLP central. Além disso, é necessário

conceber uma codificação no simulador, de modo que seja possível a comunicação com o AD e a simulação orientada com base nos eventos externos.

Como os ambientes de simulação, ambiente dinâmico e ambiente real utilizam recursos computacionais diferentes é necessário estabelecer um canal de comunicação entre esses ambientes a fim de garantir que a troca de comandos ocorra de forma semelhante a um ambiente completamente real.

O canal de comunicação entre o ambiente dinâmico e o ambiente real é descrito em Diogo (2008), sendo que para esse canal são usados os recursos do próprio *software* que emula o ambiente dinâmico.

Em relação à comunicação entre o ambiente de simulação e o ambiente dinâmico foi necessário conceber um nível adicional para que possa ser compatibilizada e uniformizada a troca de comandos e respostas. Essa interface foi concebida para poder auxiliar a comunicação e integração entre ambientes que utilizam plataformas computacionais diferentes.

3.2 Ambiente de Simulação

O ambiente de simulação é concebido para permitir a implementação do nível de procedimentos operacionais e nível de sistemas físicos a serem simulados em um *software* específico para simulação. Desta forma, neste ambiente os Procedimentos Operacionais e os Sistemas Físicos serão modelados e implementados dentro da estrutura de controle supervisório, conforme Figura 3.2.

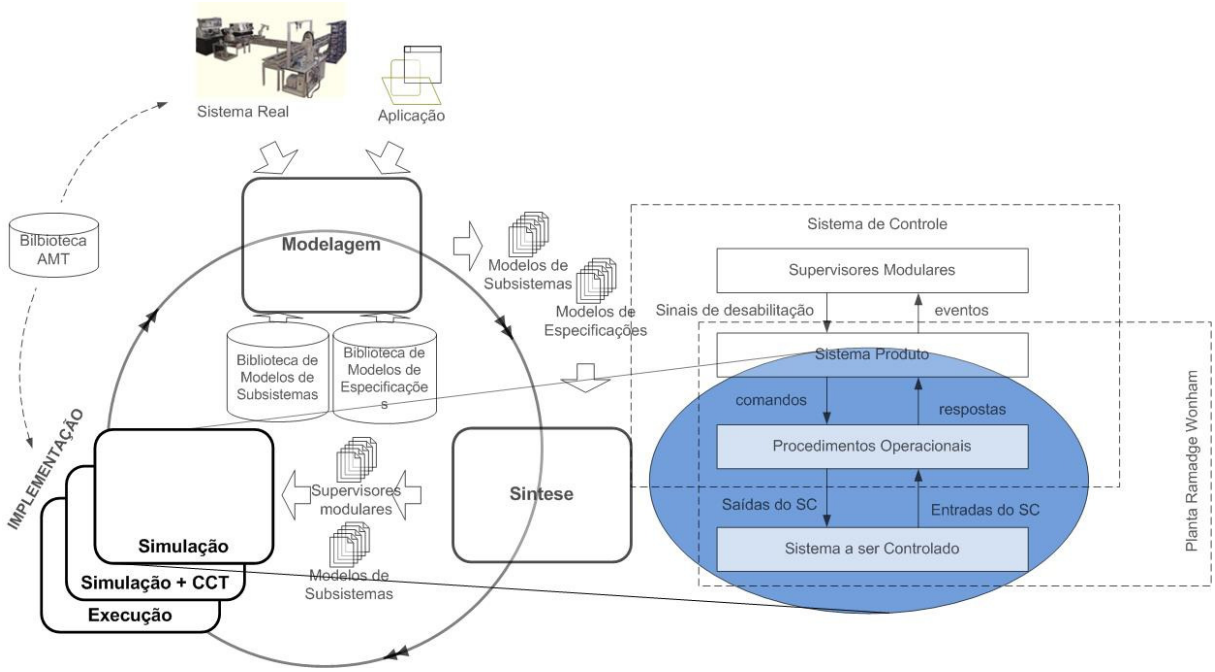


Figura 3.2 –Detalhamento do ambiente de simulação

Portanto, baseado em uma arquitetura dividida em níveis, uma adaptação do modelo proposto por Hibino et al. (2006) foi inserida. Assim, a simulação foi feita em três níveis, de acordo com a Figura 3.3.

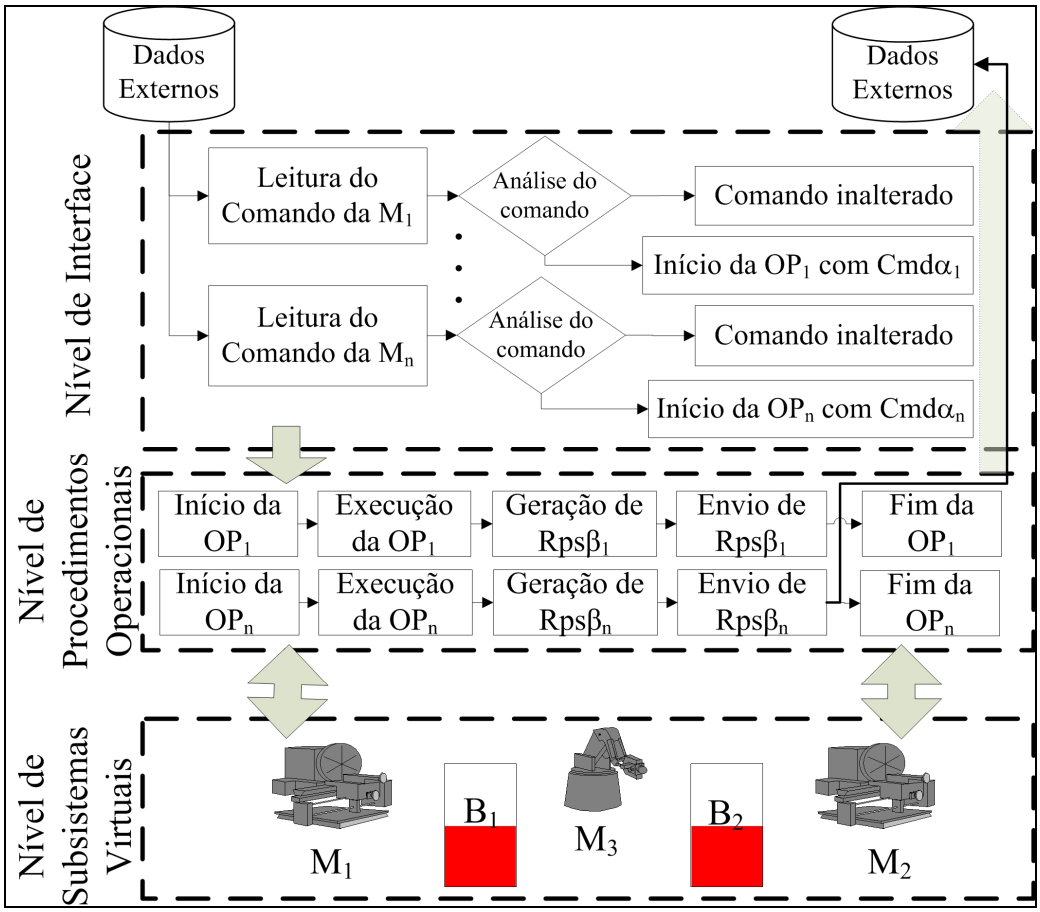


Figura 3.3 – Níveis do ambiente de simulação

O nível de interface é responsável pela comunicação entre o ambiente de simulação e o AD. Seu principal objetivo é realizar a leitura dos comandos enviados pelo AD, processá-los e executar o PO relacionado a cada comando. Os comandos enviados a partir do CLP ao AD são armazenados em um BD, e lidos pelo simulador, posteriormente. Também no simulador, uma estrutura de tomada de decisão é definida de acordo com um determinado valor lógico. Neste caso, o nível de interface envia um sinal para o nível de PO; caso contrário, espera por uma nova leitura ao BD.

Neste nível são escritas todas as rotinas que irão garantir a leitura dos comandos e envio das repostas de forma a garantir a troca de informações em tempo suficiente e seqüenciamento correto para que o ambiente dinâmico possa executar suas tarefas sem perda outro troca de informações.

Além desse fato o nível de interface é utilizado como uma camada que irá permitir a integração de ambientes de desenvolvimentos diferentes. Em muitos casos é previsto que o a aplicação a qual o ambiente dinâmico esta

implementada não seja compatível diretamente com a aplicação de simulação, assim esse nível terá a responsabilidade de resolver essa incompatibilidade.

É no nível de procedimentos operacionais que as seqüências de execuções das tarefas da planta são implementadas. Nesse nível o projetista irá desenvolver conforme os modelos previamente gerados na etapa de síntese. Todas as considerações de tempo por processo, utilização de recursos e demais restrições do modelo devem ser prevista e inseridas neste nível.

O nível PO também é responsável por enviar os comandos do nível de interface ao nível de modelos virtuais. A fim de minimizar o tempo dos dados, o evento correspondente a uma resposta é enviado diretamente ao AD pelo BD, sem ser tratado pelo nível de interface. Fica a cargo deste nível a geração dos registros, que devem conter dados quantitativos sobre a planta e sobre a capacidade produtiva. Assim, além de ser possível a validação da estrutura de controle, a decisão sobre os recursos disponíveis para planejamento da produção pode ser tomada.

O nível dos Modelos Virtuais é responsável pela simulação dos subsistemas que compõem a planta em estudo. Nesse caso, esse nível apresenta uma comunicação direta com o nível de PO e troca comandos e respostas com ele. Além disso, os modelos virtuais são implementados levando-se em conta a disposição física do modelo, tanto para a planta real como para a virtual. Dessa forma, para cada modelo de subsistema virtual há um subsistema real correspondente.

De acordo com a etapa de implementação proposta, os PO podem ser simulados um a um, como mostrado na Figura 3.4. Primeiramente, somente um modelo virtual; depois, novos modelos são progressivamente agregados até atingir a simulação completa do sistema. Ao se fazer isso, é possível determinar gradativamente se a ação de controle imposta pelo CLP está correta, ou se o projetista conseguiu atingir as especificações plenamente. A inserção progressiva dos PO permite uma verificação sistemática da síntese do SC bem como da confiança nos subsistemas inseridos.

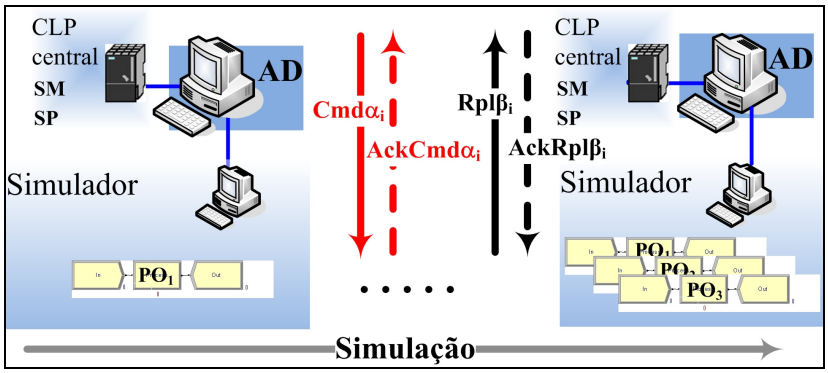


Figura 3.4 – Evolução da simulação, Diogo et al (2007).

A abordagem apresenta as seguintes vantagens sob a implementação direta em um ambiente real sem simulação:

• não é necessário desenvolver uma metodologia para tradução dos autômatos dos Supervisores Modulares e do Sistema-produto para um ambiente de simulação;

• é possível implementar e testar todos os Procedimentos Operacionais simultaneamente com os SMs e SPs já implementados num sistema real;

• garante agilidade na implementação real dos Procedimentos Operacionais: podem ser implementados no sistema real sem que todos os Procedimentos Operacionais estão fisicamente implementados;

• mantém um banco de dados contendo os Procedimentos Operacionais simulados para futuras utilizações, implementações e alterações (aproveitamento de código);

• permite a extração de dados dos processos de manufaturas real, visto que o sistema de simulação continua presente.

3.3 Interface

O nível de interface de controle de simulação é composto pelas tecnologias de comunicação entre o Ambiente de Simulação e o CLP, bem como

pelo Ambiente Dinâmico, responsável pelo controle de trocas de comandos e repostas e determinação dos subsistemas reais ou simulados.

Como um requisito definido para este trabalho é de possibilitar um suporte a implementação da TCS da forma mais abrangente possível, levantou-se um mínimo de condições para que essa implamentação seja viável.

Como nesse modelo a implementação dos dois últimos níveis da estrutura de controle adicionados do nível de interface é feita num *software* de simulação,.esse *software* deve permitir que ambos os níveis sejam implementados de acordo com a sistemática proposta acrescidos dos controle necessários para o nível de interface.

Para tanto, o *software* de simulação deve conter pelo menos

1) possibilidade de comunicação com aplicações externas;
2) possibilidade de implementação, por meio de blocos, de procedimentos operacionais;
3) possibilidade de criação de recursos físicos virtuais;
4) uma linguagem de programação capaz de ordenar a simulação conforme descrito nas etapas de simulação.

4 ETAPA DE IMPLEMENTAÇÃO

Este capítulo descreve os procedimentos para a implementação do ambiente de suporte a simulação da TCS. Nele as fases desta etapa são detalhadas de acordo com a proposta de trabalho indicada no capítulo anterior. Portanto, como resultado deste estudo, no final deste capítulo será possível desenvolver um ambiente de simulação que dará apoio a implementação da TCS.

4.1 Fase de simulação

• Coleta de dados e estabelecimento do canal de comunicação:

Nessa fase são identificados todos os comandos e repostas, bem como os subsistemas e seus respectivos Procedimentos Operacionais e Sistemas Físicos que devem ser modelados.

Faz-se necessário, em primeiro lugar, levantar os dados dos subsistemas, identificando quais são os Procedimentos Operacionais pertinentes a cada um. Desse modo, deve-se criar uma tabela que é denominada de Matriz de Subsistemas e Procedimentos Operacionais, conforme sugestão dada na Tabela 4.1.

SUBSISTEMA		**PROCEDIMENTO**	
Nome	**Identificador**	**Nome**	**Identificador**
Subsistema X	SubSisX	Procedimento Operacional X	OPX
Subsistema Y	SubSisY	Procedimento Operacional Y	OPY
Subsistema Z	SubSisZ	Procedimento Operacional Z	OPZ

Tabela 4.1– Matriz de subsistemas e procedimentos operacionais

Essa matriz serve de referência para saber quais são os subsistemas e Procedimentos Operacionais simulados e, posteriormente, migrados para o ambiente real.

Depois dessa definição, é necessário identificar os comandos e as respostas que são trocados com o Ambiente Dinâmico por meio das tecnologias de comunicação. Para que seja preservada a estrutura de controle e melhorada sua organização, é sugerido ordenar os comandos (cmd), reconhecimentos de leitura de comandos (ackread), procedimentos operacionais (op), respostas (resp) e informação de procedimento simulado ou executado (real) da seguinte forma:

- cmdα, em que α corresponde a um comando gerado pelo Sistema-produto
- ackreadα, em que α corresponde a uma informação de reconhecimento de leitura de comando
- opα, em que α corresponde a um procedimento operacional a ser simulados
- respα, em que α corresponde a uma reposta gerada pelos Procedimentos Operacionais
- realα, em que α corresponde a um Procedimento operacional a ser transportados do ambiente simulado para o ambiente real.

Sendo assim pode-se montar a Matriz de Comandos e Repostas da forma mostrada na Tabela 4.2.

	COMANDO	RESPOSTA	PROCEDIMENTO
	cmdαX	rplβX	OPX
	cmdαY	rplβY	OPY
	cmdαZ	rplβZ	OPZ
		rplμZ	OPZ

Tabela 4.2– Matriz de Comandos e Repostas

Por causa da possível incompatibilidade entre os *softwares* de simulação e os *softwares* de controle e aquisição de dados dos CLPs nos quais estarão os Sistemas Produto, é necessário implementar um arquivo de controle intermediário a esses dois *softwares* para possibilitar a comunicação e controlar o tráfego das informações.

O próximo passo é montar um *layout* de arquivo de comunicação entre os *softwares* de simulação e o sistema representante do Ambiente Dinâmico.

Esse *layout* deve apresentar uma matriz M_{ij} , em que os **i** são os subsistemas dos Procedimentos Operacionais e os **j**, os comandos, respostas e a

variável que irá definir se o subsistema está sendo simulado ou já foi implantado no ambiente real, conforme Figura 4.1.

	Real	cmdj	rplj	ackcmdj	ackrplj	cmdj	rplj
OPi	0	0	0	0	0	0	0
OPi	0	0	0	0	0	0	0
OPi	0	0	0	0	0	0	0

Figura 4.1– *Layout* do arquivo de controle de comunicação

Depois de levantar esses dados, é necessário referenciar as variáveis dos sistemas de simulação e o *layout* do arquivo de controle de comunicação, para que seja possível programar ambos os *softwares*, mantendo as identidades e padrões necessários que evitam conflito entre comandos e repostas e as diversas variáveis do sistema.

A Tabela 4.3 apresenta esse relacionamento para o caso de sua implementação no *software* Microsoft Excel.

Comando/Resposta/Controle	Simulador	Scada
cmdαX	C3	L3C3
cmdαY	C4	L4C3
cmdαZ	C5	L5C3
rplβX	D3	L3C4
rplβY	D4	L4C4
rplβZ	D5	L5C4
rplμZ	H5	L5C4
*ackcmd*αY	E3	L3C5
*ackcmd*αY	E4	L4C5
*Ackcmd*αZ	E5	L5C5
*ackrpl*βX	F3	L3C6
*ackrpl*βY	F4	L4C6
*ackrpl*βZ	F5	L5C6
*ackrpl*μZ	H5	L5C6
OPX	B3	L3C2
OPX	B4	L4C2
OPX	B5	L5C2

Tabela 4.3 – Relacionamento entre a matriz de comunicação e as variáveis e apontamentos nos *software*

• **Modelos no ambiente de simulação**

Com todos os dados já levantados, é possível iniciar a modelagem no ambiente de simulação, que deve prever a implementação dos modelos nos três níveis propostos.

Como sugestão, o primeiro nível a ser modelado deve ser o nível de Procedimentos Operacionais. Isso porque em um nível de interface o simulador irá utilizar as variáveis criadas no nível de procedimentos operacionais e, para isso, o já deve estar implementado.

Para modelar este nível é necessário que sejam previstos quatro itens:

• a execução de cada procedimento operacional, que deve respeitar as especificações do modelo proposto;

• o ponto de validação, por meio de chamada ao nível de interface, que valida o procedimento operacional que está sendo executado, avaliando se está sendo somente simulado ou simulado e também executado no ambiente real;

• um ou mais pontos para devolução das respostas, caso o procedimento operacional esteja sendo somente simulado;

• trocas de comandos com o nível de Interface e com o nível de subsistema físico virtual.

Para o nível de subsistemas Físico Virtual devem ser implementados os modelos dos recursos físicos que obedeçam suas características. Esse nível também deve prever a troca das entradas e saídas do Sistema de Controle.

• **Programação do controle do nível de Interface**

Nessa fase são definidas as rotinas para a troca dos comandos entre o Ambiente de Simulação e o *Software* SCADA e também a implementação do modelo do Procedimento Operacional.

• **Troca dos comandos**

A troca de comandos deve assegurar ao simulador ler sempre de forma seqüencial o comando dado, para impedir a leitura de um mesmo comando duas vezes ou mais. Para que isso ocorra e para que a troca de comandos evolua, determinam-se as seguintes funções de controles, definidas nas figuras Figura 4.2, Figura 4.3 e Figura 4.4, para a implementação no *software* de simulação:

```
Blocc  Inicializa Variáveis
//inicialização das Variáveis
//declaração das variáveis
varOpn como inteiro
cmdαn como inteiro
rplβn como inteiro
ackcmdαn como inteiro
ackrplβn como inteiro

//Definindo os valores iniciais das variáveis
//varOpn  será definida pelo Ambiente Dinâmico
cmdα := 0
rplβn  := 0
ackcmdαn  := 0
ackrplβn  := 0
//término de inicialização das variáveis
```

Figura 4.2 – Função de definição e inicialização das variáveis

```
Funçãc  Verificação de gravação de comando
//Inicio da função
Lê variável ackreadαn
Se ackreadαn := 0 então
    Chama função leitura do comando
Se não
    Define que não será executado nenhum Procedimento Operacional
Fim Se
//Fim da função
```

Figura 4.3 – Função leitura e verificação de gravação de comando

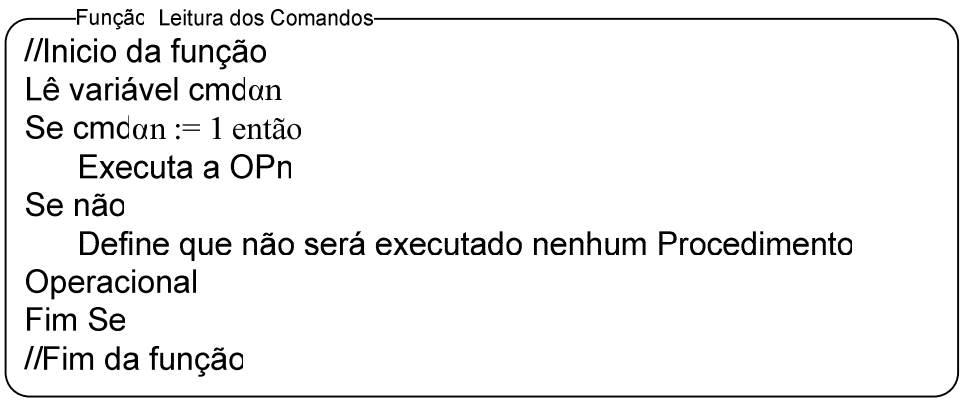


Figura 4.4- Função leitura dos comandos

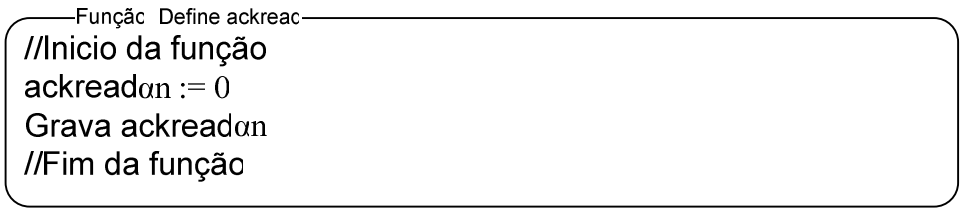


Figura 4.5- Função define *ackread*

A fim de garantir o correto seqüenciamento dos comandos, as funções descritas anteriormente devem respeitar o fluxo de controle descrito na Figura 4.6.

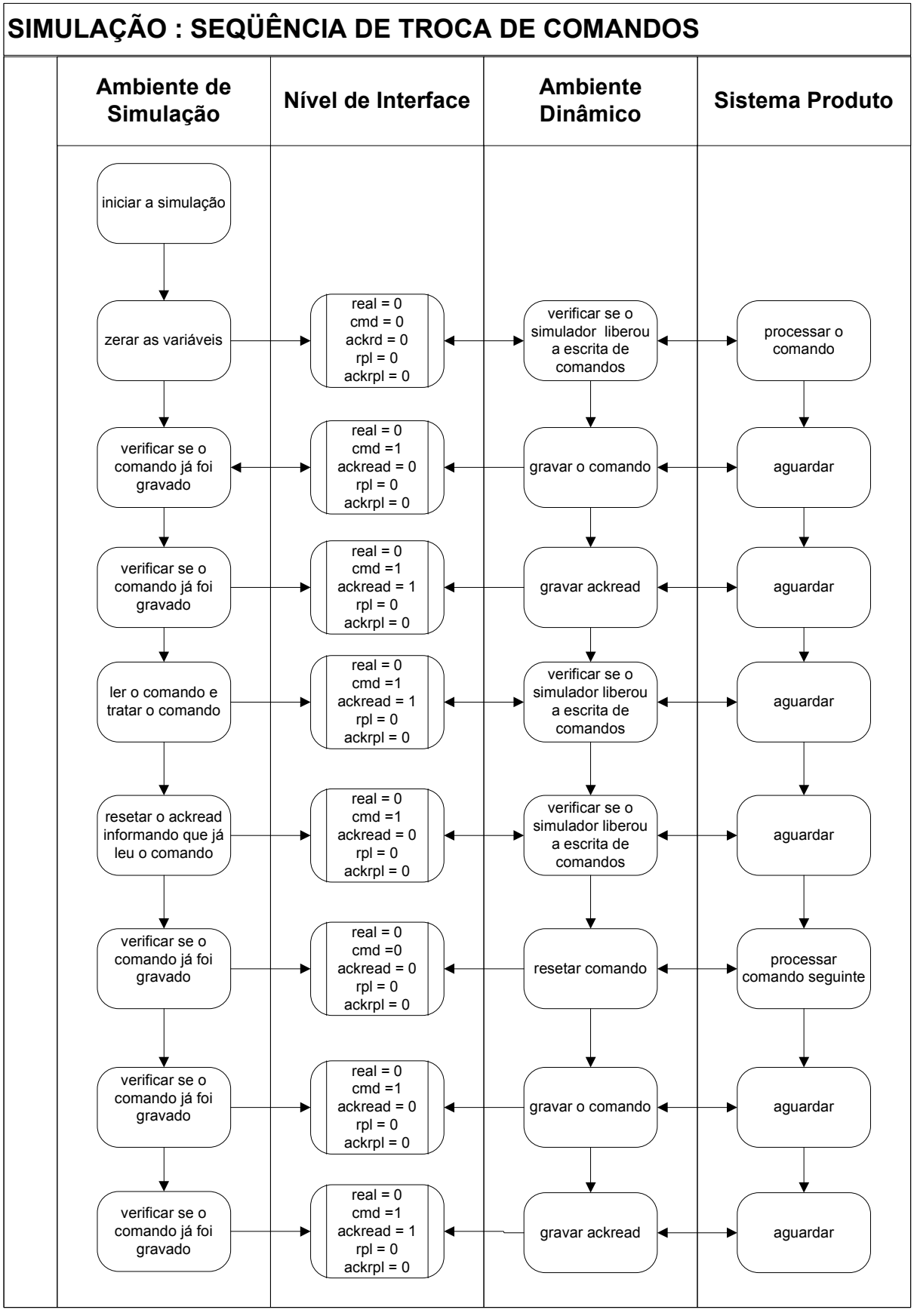


Figura 4.6 – Fluxo de controle de troca dos comandos

• **Troca de respostas**

A troca de respostas deve assegurar ao simulador ler as respostas sempre de forma seqüencial e incremental.

Para que isso ocorra e para que a troca de resposta evolua, determinam-se as seguintes funções de controles, definidas nas figuras 4.12, 4.13 e 4.14.

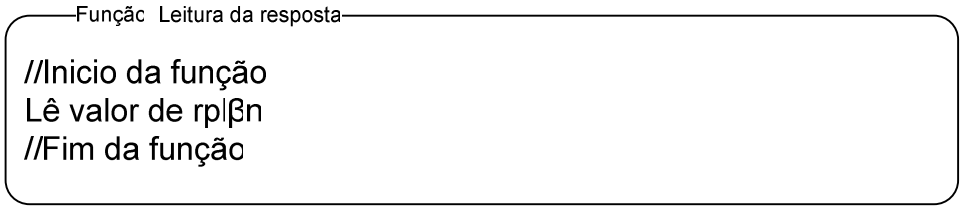


Figura 4.7 – Função leitura da resposta

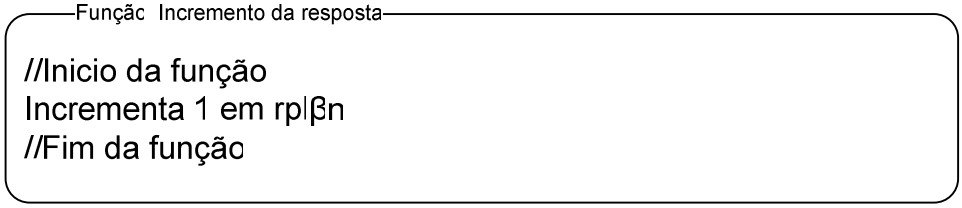


Figura 4.8 – Função incremento da resposta

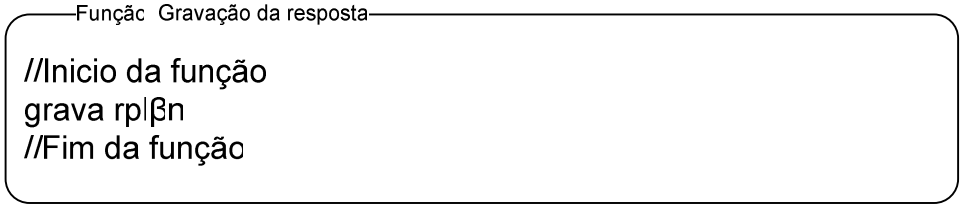


Figura 4.9 – Função gravação da resposta

Da mesma forma que para o controle dos comandos, existe a necessidade de seguir o fluxo de controle das respostas para a implementação das funções anteriormente descritas. O fluxo de controle das respostas pode ser observado na Figura 4.10.

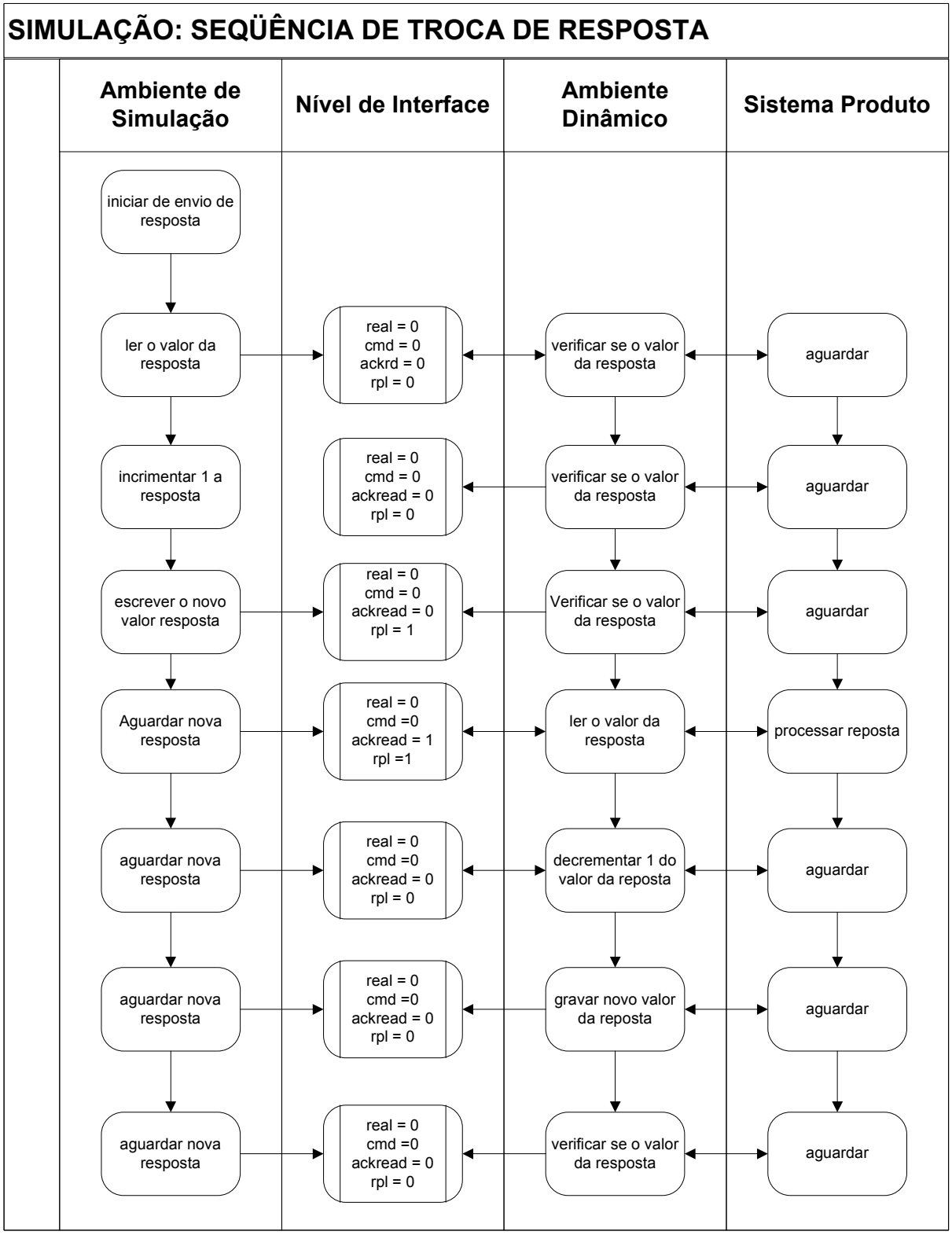


Figura 4.10 – Fluxo de controle de troca das respostas

A implementação feita desta forma visa a garantir o seqüenciamento correto na troca dos comandos e respostas, evitando que eles sejam alterados sem terem sido previamente tratados pelo simulador e pelo Ambiente Dinâmico.

Essa garantia é dada por meio de um conjunto de variáveis auxiliares denominadas de *ackread* e *ackrpl*, que apresentam as informações de reconhecimento de envio e leitura dos comandos e respostas. Sendo assim, quando um comando novo é enviado pelo Ambiente Dinâmico, a variável *ackcmd* é setada para zero e, desse modo, quando o simulador efetuar seu ciclo de leitura, irá detectar que aquele comando presente é um novo comando, processando-o. Uma vez que esse comando seja interpretado pelo simulador, ele deve restabelecer o valor dessa variável para um, liberando o Ambiente dinâmico do envio de um novo comando.

O mesmo ocorre no envio das respostas, por meio do *ackrpl*. Porém, neste caso, como os eventos gerados são eventos não controláveis, o retorno das respostas do simulador ao ambiente dinâmico é dado por meio de um incremental unitário na variável *ackrpl*, somando-se um ao valor atual da variável toda vez que uma nova resposta for enviada. Quando o Ambiente Dinâmico efetuar o seu ciclo de leitura, irá verificar o valor desta variável e decrementar um; a resposta então será tratada.

Essa forma de implementação confere ao ambiente uma confiabilidade na troca dos comandos e respostas, garantindo que não será enviado um segundo comando sem que o anterior já tenha sido interpretado pelo ambiente de simulação. Desta forma, percebe-se que, quanto mais rápidas forem as leituras do ambiente de simulação, mas rápido será liberado o Ambiente Dinâmico de efetuar outro envio de comando. Isso permite uma aproximação de execução da simulação em tempo real, bastando para isso que os ciclos de leitura e gravação do ambiente de simulação sejam inferiores aos ciclos de leitura e gravação do Ambiente Dinâmico.

4.2 Fase de simulação + CCT

Esta é a fase na qual os subsistemas são gradativamente migrados para a o ambiente real, o que permite que seja possível a implementação de Procedimentos Operacionais em um ambiente real, conforme possam ser validados em conjunto com o restante da planta que ainda está sendo simulada.

Para efeitos de comparação de resultados, o ambiente simulado deve manter o Procedimento Operacional migrado. Desta forma, o ambiente simulado e o ambiente real podem ser visualizados conforme Figura 4.11.

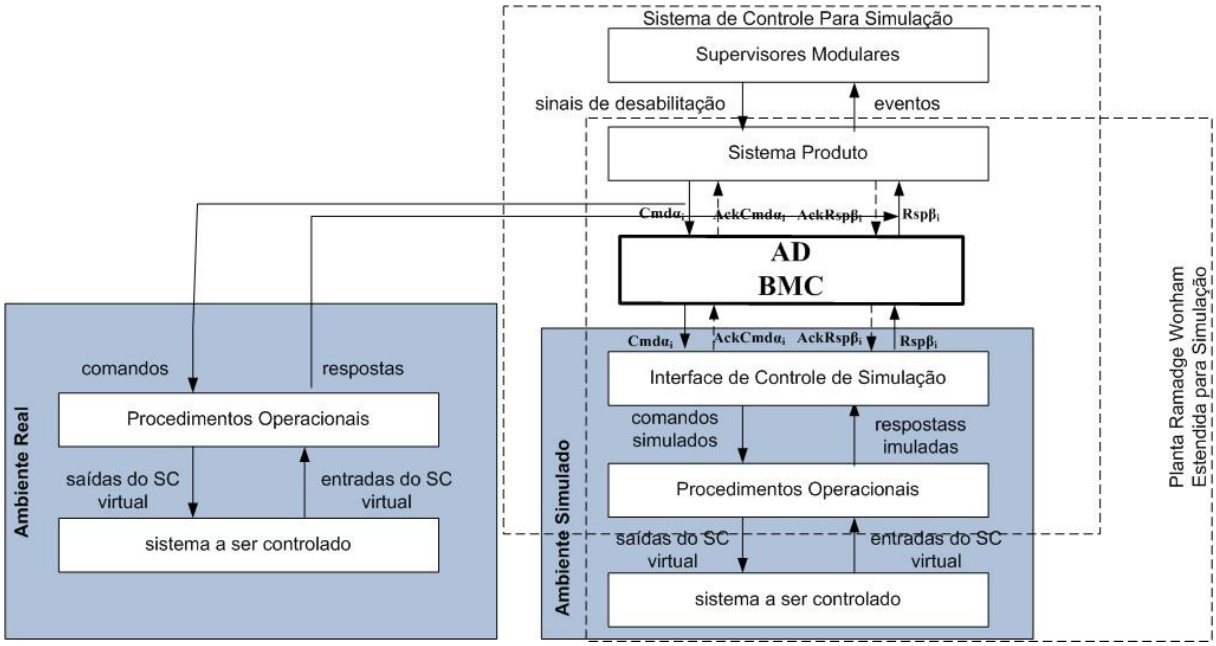


Figura 4.11 – Ambiente simulado + ambiente real

Há, então, a necessidade de o nível de interface controlar a informação de comandos e respostas, e os procedimentos operacionais devem saber tratar quais respostas devem ser devolvidas ao ambiente real.

Para isso, no nível de interface, a informação de controle que informa que determinado OP está sendo executado no ambiente real deve ser alterada e sua variável habilitada, ou seja, seu valor deverá ser um. Na Figura 4.12 é apresentado um exemplo em que o OP1 está sendo executado no ambiente real e os outros OPs estão sendo simulados.

	A	B	C	D	E	F	G	H
1					Arena			
2		Real	cmd1	rpl1	ackcmd1	ackrpl1	rpl2	ackrpl2
3	OP1	1	0	0	0	0	0	0
4	OP2	0	0	0	0	0	0	0
5	OP3	0	0	0	0	0	0	0

Figura 4.12 – Arquivo de controle de comunicação com OP executada ambiente real e virtual

As funções relativas às leituras e aos tratamentos dos comandos permanecem inalteradas, visto que o processo de leitura dos comandos que são

destinados ao ambiente real deve ser feito pelo simulador. Dessa forma, todo comando executado no ambiente real é também simulado e os dados de comparação podem ser extraídos visualmente e em relatórios.

As funções relativas às gravações das respostas também permanecem inalteradas, já que elas também devem ser tratas na simulação para efeitos de comparação e registro. Porém, a função de gravação será alterada a fim de impedir que as repostas para o procedimento operacional executado no ambiente simulado sejam passadas para o ambiente dinâmico como sendo as repostas executadas no ambiente real, conforme Figura 4.13.

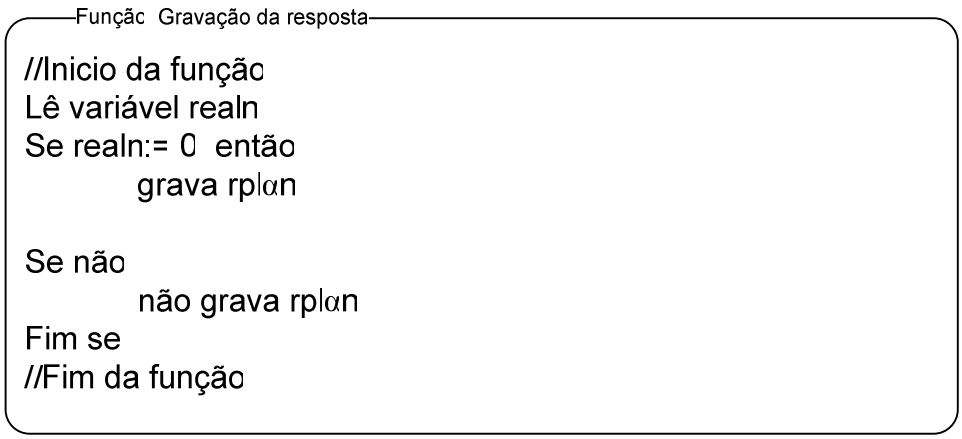


Figura 4.13 – Função gravação da resposta

A simulação +CCT pode ser implementada de acordo com o fluxo de controle de comando, Figura 4.14 e fluxo de controle de resposta, Figura 4.15.

Com a implementação dessa estrutura, composta por esses blocos de comandos e estrutura de controle, não se faz mais necessário alterar o ambiente desenvolvido, e, conforme a implantação proposta por Diogo et al (2008), o ambiente dinâmico pode controlar, por meio da alteração da variável de controle de ambiente, quais serão os Procedimentos Operacionais que serão simulados ou executados.

Assim, todos os procedimentos operacionais podem ser simulados na íntegra, ou executados parcialmente em um ambiente real e parcialmente simulados em um ambiente virtual, trocando informações, em que podem ser estraídos dados e relatórios no ambiente de simulação.

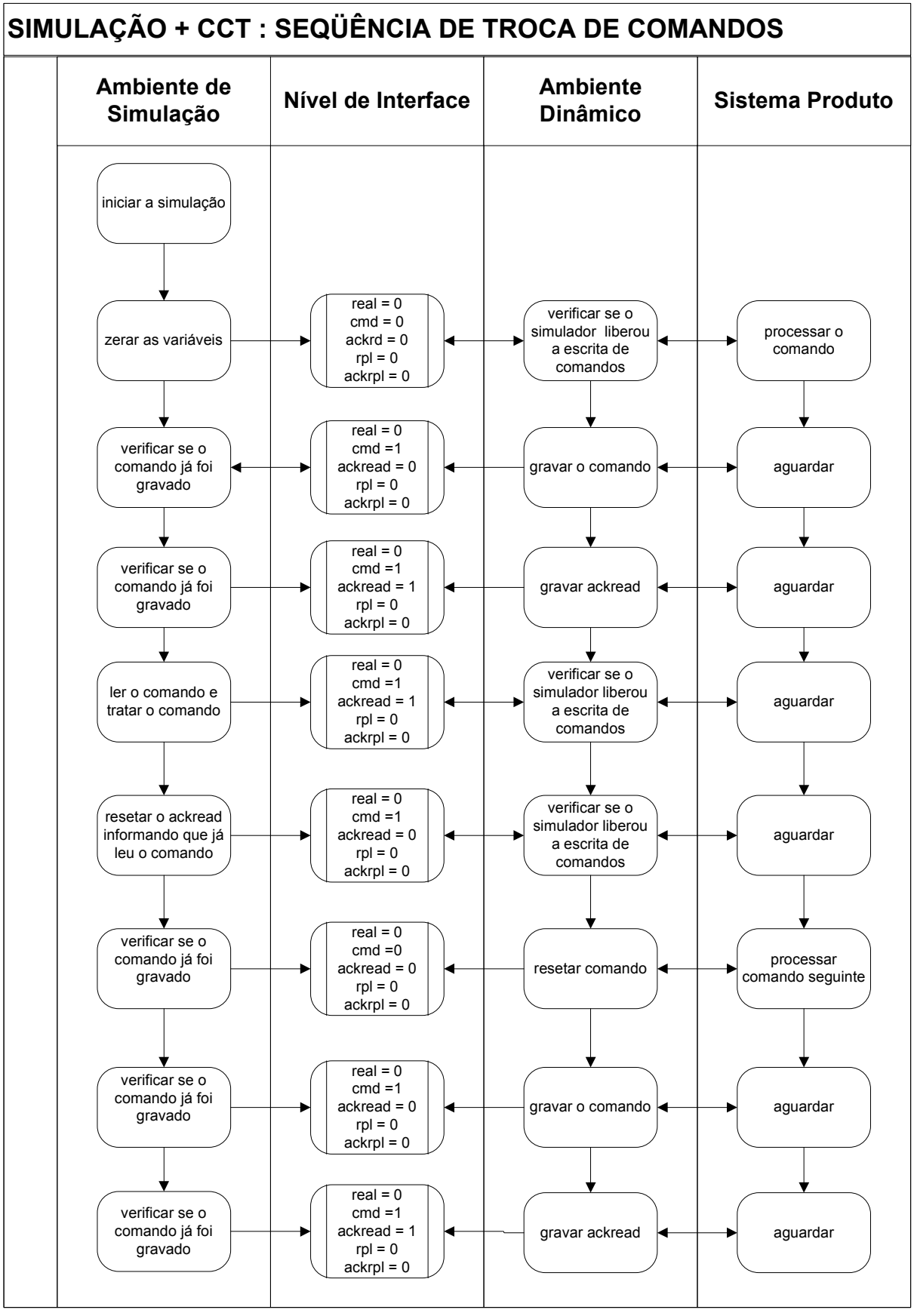


Figura 4.14 – Fluxo de controle de comandos para o Ambiente de Simulação + CCT

Figura 4.15 – Fluxo de controle de respostas para o Ambiente de Simulação + CCT

4.3 Fase de execução

A fase de execução é a última da etapa de implantação do Ciclo de Desenvolvimento proposto por Busseti e Portela (2006), e ocorre quando todos os procedimentos operacionais já estão implementados no ambiente real, sendo necessário somente marcar a variável que define em qual ambiente está sendo executado o procedimento operacional para o ambiente real para todos os OP.

O ambiente de simulação, por meio do recebimento dos comandos do ambiente dinâmico, atua para efeitos de comparação e análise do sistema real implementado, somente.

5 EXEMPLO DE APLICAÇÃO

Neste capítulo é apresentado um exemplo de aplicação da sistemática de simulação proposta, a fim de validar e testar as funcionalidades descritas. Neste exemplo foram implementados os níveis de supervisores e sistema produto num CLP real e através de atuadores foram gerados os eventos não controláveis de forma aleatório e independente. Os níveis de procedimentos operacionais e sistemas físicos virtuais inicialmente foram implementados no ambiente de simulação e posteriormente implementados no CLP real para validar as fases da etapa de simulação. Nos itens abaixo são descritos os procedimentos adotados para essa implementação.

5.1 Definição do ambiente de estudo de caso

Nesse estudo de caso, implementa-se a sistemática de simulação em um sistema de manufatura composto por três máquinas e dois *buffers*, um entre as máquinas 1 e 3 e outro entre as máquinas 3 e 2, conforme exemplificado na Figura 5.1. As máquinas são representadas por M1,M2 e M3 e os buffers por B1 e B2, unitários, com capacidade de armazenamento igual a um.

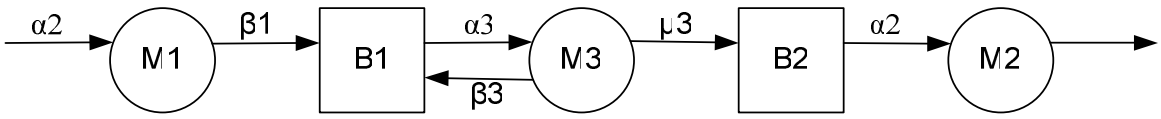


Figura 5.1 – Sistema de manufatura de três máquinas

Este exemplo tem como base a distribuição de estrutura de controle Supervisório de Sistemas a Eventos Discretos proposta por Viera el al (2004). Em seu trabalho, Vieira el al (2004) usa este sistema de manufatura com a finalidade de exemplificar a aplicação da distribuição da estrutura de controle em supervisores,

sendo essa a abordagem adotada nessa implementação. Com este exemplo pode-se validar e assegurar a viabilidade de implementação, a modularidade do sistema físico e validar a reutilização das estruturas de controle previamente implementadas.

Os geradores associados às máquinas para esses sistemas são definidos na Figura 5.2, em que M1 no estado 0 significa máquina em repouso e no estado 1 significa máquina trabalhando; o mesmo ocorre para M2. Para M3, o estado 0 significa que o manipulador robótico esta em repouso, o estado 1 significa que o manipulador está levando a peça do *buffer* B1 para o *buffer* B2 e no estado 2 significa que o manipulador está retornando ao estado de repouso.

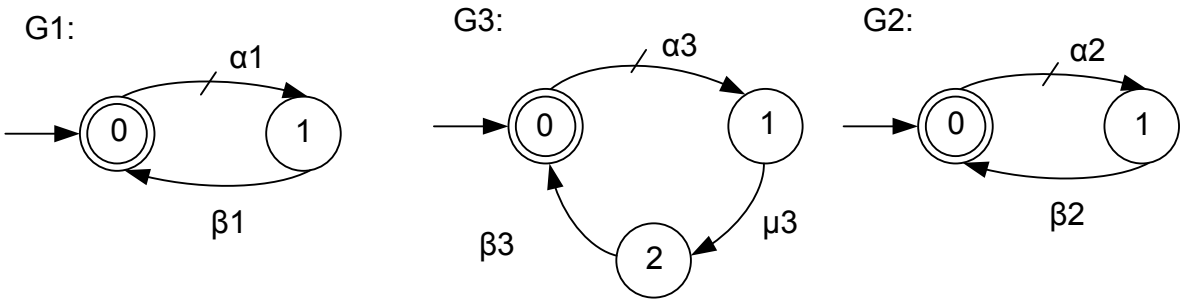


Figura 5.2 – Geradores das máquinas M1, M2 e M3

Para este caso, identificam-se os eventos **α** (início de operação) e **β** (fim de operação) como eventos controláveis e não-controláveis, respectivamente.

Como o objetivo da Teoria de Controle Supervisório é o de utilizar ao máximo os recursos disponíveis e de forma correta, garantindo a segurança do procedimento, deve-se atentar a alguns detalhes que resultam em algumas especificações. Ao verificar o sistema proposto, observa-se que ele não pode causar *underflow* nem *overflow* nos *buffers*, ou seja, não pode haver sobreposição de peças e as máquinas não podem retirar peças de um *buffer* vazio. Essas duas especificações, E1 e E2, são representadas pelos geradores da Figura 5.3.

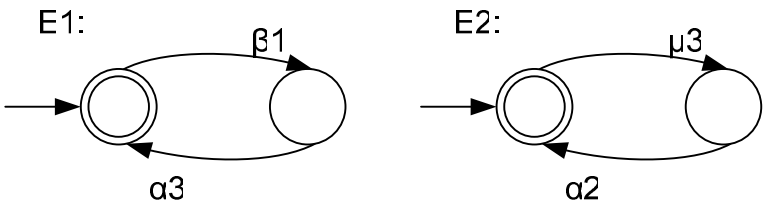


Figura 5.3 – Especificações de controle E1 e E2

A composição dessas especificações evita que ocorram o *underflow* e o *overflow*.

Pela síntese dos supervisores modulares reduzidos, conforme Queiroz e Cury (2000), são obtidos os supervisores para esse sistema de manufatura mostrado na Figura 5.4.

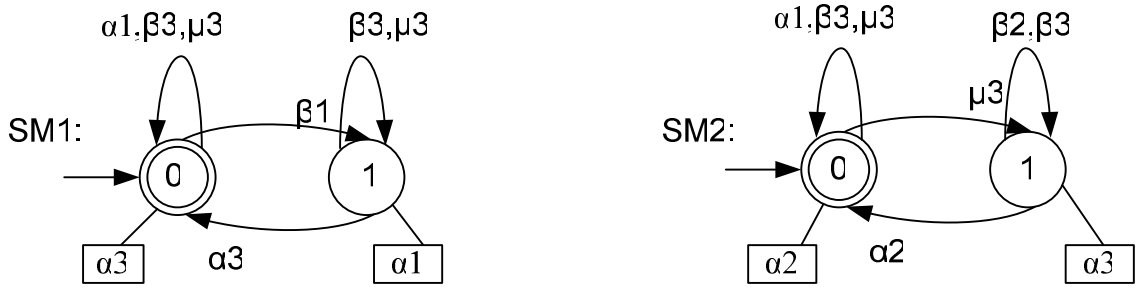


Figura 5.4 – Supervisores modulares para o sistema de manufatura

Observa-se, na Figura 5.4, que os eventos nos arcos de transição são os eventos habilitados e os eventos dentro dos quadrados são os eventos desabilitados pelo supervisor e desta forma não ocorrerem naquele estado.

Na figura 5.5 são apresentados os diagramas funcionais dos procedimentos operacionais deste sistema de manufatura, conforme IEC61131-3, 1993.

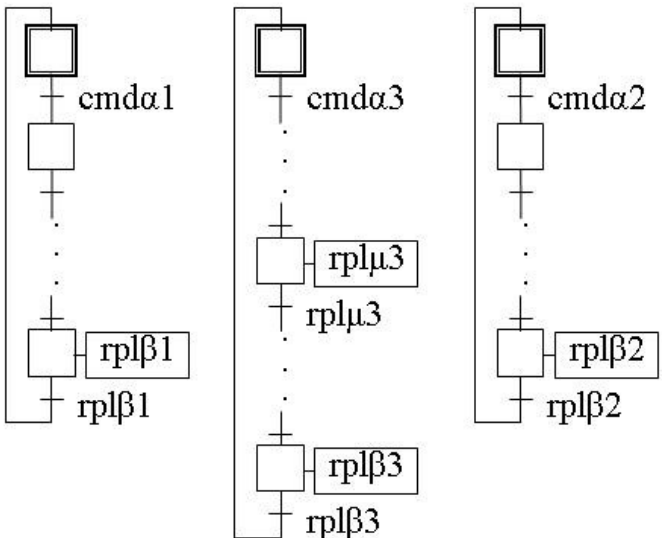


Figura 5.5 – Diagramas funcionais dos Procedimentos Operacionais do sistema de manufatura

Segundo Viera e Cury (2004), um procedimento operacional é iniciado no momento em que ele recebe um comando cmdαi , neste caso os comandos recebidos pelo Sistema-produto iniciam as operações das máquinas, que ao terminarem seus procedimentos operacionais geram uma resposta rplβi ou rplμi que esta relacionada a um evento não controlável.

5.2 Levantamento de informações do ambiente de simulação

5.1.2 Coleta de dados e estabelecimento do canal de comunicação

Para o exemplo em questão, considera-se a arquitetura de controle inicialmente composta por um CLP na qual estão implementados os níveis de Supervisores e o de Sistema-produto. Esse CLP é chamado de CLP0 e é implementado em um simulador comercial, que neste caso é o Arena, usado para a implantação dos níveis de Procedimento operacional simulado e sistema físico simulado. Na seqüência, quando os procedimentos operacionais forem migrados para os ambientes reais, é inserido um CLP para cada um, e o OP1 implementado no CLP1, o OP2 no CLP2 e o OP3 no CLP3

Como primeira etapa, devem ser identificados os números de subsistemas e seus respectivos procedimentos operacionais, para que sejam obtidos conforme Tabela 5.1.

SUBSISTEMA		**PROCEDIMENTO**	
Nome	**Identificador**	**Nome**	**Identificador**
Subsistema 1	SubSis1	Procedimento Operacional 1	OP1
Subsistema 2	SubSis2	Procedimento Operacional 2	OP2
Subsistema 3	SubSis3	Procedimento Operacional 3	OP3

Tabela 5.1 – Matriz subsistemas e seus procedimentos operacionais

Passa-se agora à enumeração de todos os comandos e respostas, identificando a quais procedimentos operacionais eles fazem parte, a fim de ser criada a matriz de troca de comandos.

Para esse exemplo define-se a matriz desta forma:

	Comando	**Resposta**	**Procedimento**
	cmdα1	rplβ1	OP1
	cmdα2	rplβ2	OP2
	cmdα3	rplβ3	OP3
		rplμ3	OP3

Tabela 5.2 – Matriz de comandos e repostas

Para cada comando e reposta é inserido um bit para controle, desta obtendo-se também a matriz de controle:

	Comando	**Resposta**	**Procedimento**
	*ackcmd*α2	*ackrpl*β1	OP1
	*ackcmd*α2	*ackrpl*β2	OP2
	*ackcmd*α3	*ackrpl*β3	OP3
		*ackrpl*μ3	OP3

Tabela 5.3 – Matriz de controle

Para complementar essas duas matrizes, pode ser definida a lista dos subsistemas que se pretende simular e controlar.

Nome do Subsistema
Subsistema 1
Subsistema 2
Subsistema 3

Tabela 5.4 – Sistemas a serem controlados

Assim, compondo as matrizes de comando e reposta, de controle e de sistemas a serem controlados, o arquivo elaborado para seguir a matriz de comunicação fica conforme a figura 5.6.

	A	B	C	D	E	F	G	H
1					Arena			
2		Real	cmd1	rpl1	ackcmd1	ackrpl1	rpl2	ackrpl2
3	OS1	0	0	0	0	0	0	0
4	OS2	0	0	0	0	0	0	0
5	OS3	0	0	0	0	0	0	0

Figura 5.6 – *Layout* do arquivo de controle de comunicação (completo.xls)

Uma vez definido o *layout*, é possível apontar as variáveis do *software* Arena e do E3 nesse arquivo, para que, posteriormente, sejam implementados os códigos-fonte. Esse apontamento é mostrado na tabela 5.5.

Comando/Resposta/Controle	**Arena**	**E3**
cmdα1	C3	L3C3
cmdα2	C4	L4C3
cmdα3	C5	L5C3
rplβ1	D3	L3C4
rplβ2	D4	L4C4
rplβ3	D5	L5C4
rplμ3	H5	L5C4
*ackcmd*α2	E3	L3C5
*ackcmd*α2	E4	L4C5
*ackcmd*α3	E5	L5C5
*ackrpl*β1	F3	L3C6
*ackrpl*β2	F4	L4C6
*ackrpl*β3	F5	L5C6
*ackrpl*μ3	H5	L5C6
OP1	B3	L3C2
OP2	B4	L4C2
OP3	B5	L5C2

Tabela 5.5 - Relacionamento entre a matriz de comunicação e as variáveis e apontamentos nos *software.*

5.2.1 Escolha das tecnologias

Antes de iniciar a programação dos níveis a serem simulados, foram escolhidas as tecnologias que foram empregadas nessa implementação, conforme

as restrições e necessidades identificadas no capítulo anterior. Para compor todo o ambiente de simulação foram usados os seguintes *softwares*:

• como *software* de simulação foi selecionado o Arena®, pois garante a implementação dos modelos dos procedimentos operacionais, dos sistemas virtuais e permite como complemento a programação na linguagem *Visual Basic Script*, estabelecendo o canal de comunicação;

• para o ambiente dinâmico é utilizado o *software* Elipse E3®, conforme proposto e implementado por Diogo et al(2008);

Como a versão do *software* Arena utilizada não consegue interagir diretamente com o Elipse E3, foi utilizado o *software* Microsoft Excel para atuar como canal de comunicação entre eles, sendo o responsável somente por facilitar a troca de comandos e repostas entre o ambiente de simulação e o dinâmico.

Uma vez que os *softwares* escolhidos operam somente na plataforma Microsoft Windows®, o sistema operacional usado foi o Microsoft Windows XP®.

5.3 Integração do ambiente de controle e comunicação com o ambiente dinâmico

Como seqüência a essa implementação, devem ser modelados e programados os níveis de sistema produto, a fim de possibilitar a criação e definição de todas as variáveis nele usadas, desta forma, foram criados os modelos dos três procedimentos operacionais, usando os blocos Station, Process e Dispose, disponíveis no *software* Arena. Uma vez os três procedimentos implementados, as variáveis foram vinculadas as inícios e fim das operações usando os blocos VBAs.

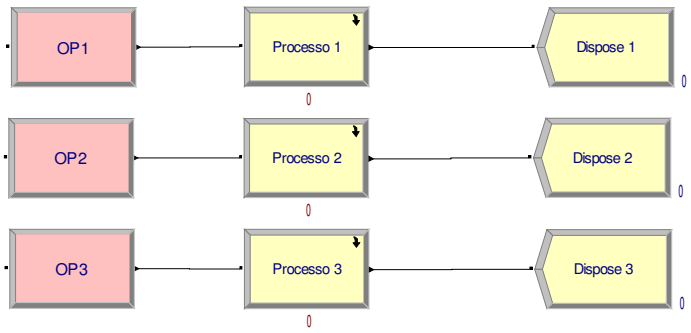
O bloco *station*, foi usado para identificar o início do procedimento operacional. Para cada um dos procedimentos foi criado um *station* com o nome correspondente: OP1, OP2 e OP3, que identificam o início dos procedimentos operacionais 1, 2 e 3, respectivamente.

Em seguida, foi inserido um bloco *Process*, obedecendo à mesma regra de nomenclatura, definindo-se o processo 1, processo 2 e processo 3, para

que cada bloco possa representar as seqüências respectivas de operações. Esses blocos foram transformados em sub-modelos para que o programador pudesse ter mais liberdade nas implementações dos procedimentos e fosse facilitada a implementação dos códigos usados.

No início de cada sub-modelo, foi implementado um bloco VBA, que verifica se o procedimento será simulado ou se já está implementado no ambiente real, além de definir a variável correspondente. Depois desse bloco, foi implementado um procedimento padrão para fins de simulação e, ao fim de cada um deles, foi inserido novamente um bloco VBA que grava a resposta dada, caso o procedimento tenha sido definido como simulado, além de gravar a confirmação de que uma resposta nova foi gerada para aquele processo.

Após o tratamento da reposta por esse bloco, um bloco *Dispose* foi inserido no final de cada Procedimento Operacional para informar o simulador sobre o término do procedimento. Desse modo, o nível de Procedimentos Operacionais ficou implementado conforme Ffigura 5.7.



Ffigura 5.7 – Nível de procedimento operacional implementado

No caso do procedimento operacional três, como são executadas duas rotinas, o sub-modelo apresenta dois blocos de processos e, ao final de cada um deles, um bloco VBA para tratar os eventos de resposta, como observado na Figura 5.8.

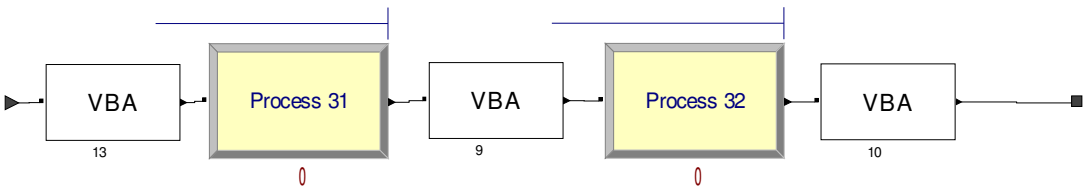


Figura 5.8 – Detalhamento do Processo 3

Depois da implementação do nível de sistemas virtuais, em que os recursos foram inseridos e os vínculos com os processos foram feitos, há os recursos de duas máquinas genéricas, que estão efetuando os procedimentos operacionais 1 e 2, um robô que executa o procedimento operacional 3 e dois *buffers* para atender as especificações do modelo, que pode ser verificada conforme Figura 5.9.

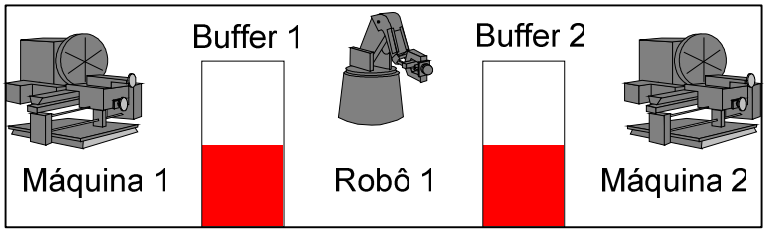


Figura 5.9 – Nível de sistemas virtuais

A implementação do nível foi feita por meio de um bloco *Create*, no qual foram definidos os intervalos de leitura dos comandos. Esse bloco, no intervalo definido, inicia a execução de um bloco VBA, que chama a rotina de leitura. Uma vez terminada essa leitura, um bloco *Route* é adicionado para que a decisão de qual procedimento operacional seja acionada.

Um conjunto formado por um bloco *Station*, chamado *idle*, e um bloco *Dispose* também foi implementado para que, caso nenhum comando seja detectado pela rotina anterior, o bloco *Route* possa tomar a decisão de não executar nada, indicando que a simulação está num estado de repouso, o que pode ser observado na Figura 5.10.

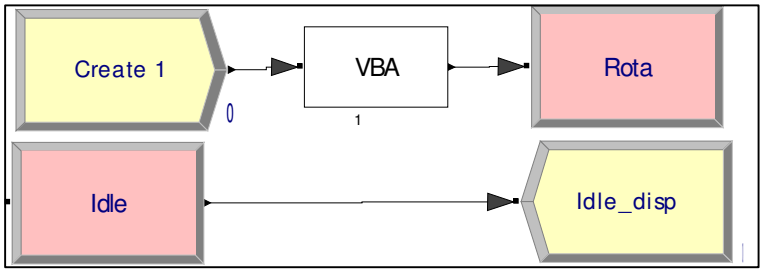


Figura 5.10 – Implementação do nível de interface

Posteriormente, foram inseridos na programação VBS todos os blocos de controle, conforme programação constante no apêndice A. Essa programação levou em consideração o uso do arquivo em Excel, conforme e a

matriz de relacionamento entre os comandos, variáveis do Arena e E3, para completar a controle da troca de comandos e respostas.

Com essa implementação, o ambiente de simulação manteve-se conectado ao ambiente dinâmico, conforme definido por Diogo et al (2008), concluindo-se a implementação do ambiente proposto por esse trabalho.

5.4 Validação da sistemática

Para validar o funcionamento da sistemática implementada, inicialmente, com os três procedimentos operacionais e com os sistemas virtuais já implementados no ambiente de simulação e no Ambiente Dinâmico, definiram-se os procedimentos operacionais como simulados, conforme Figura 5.11.

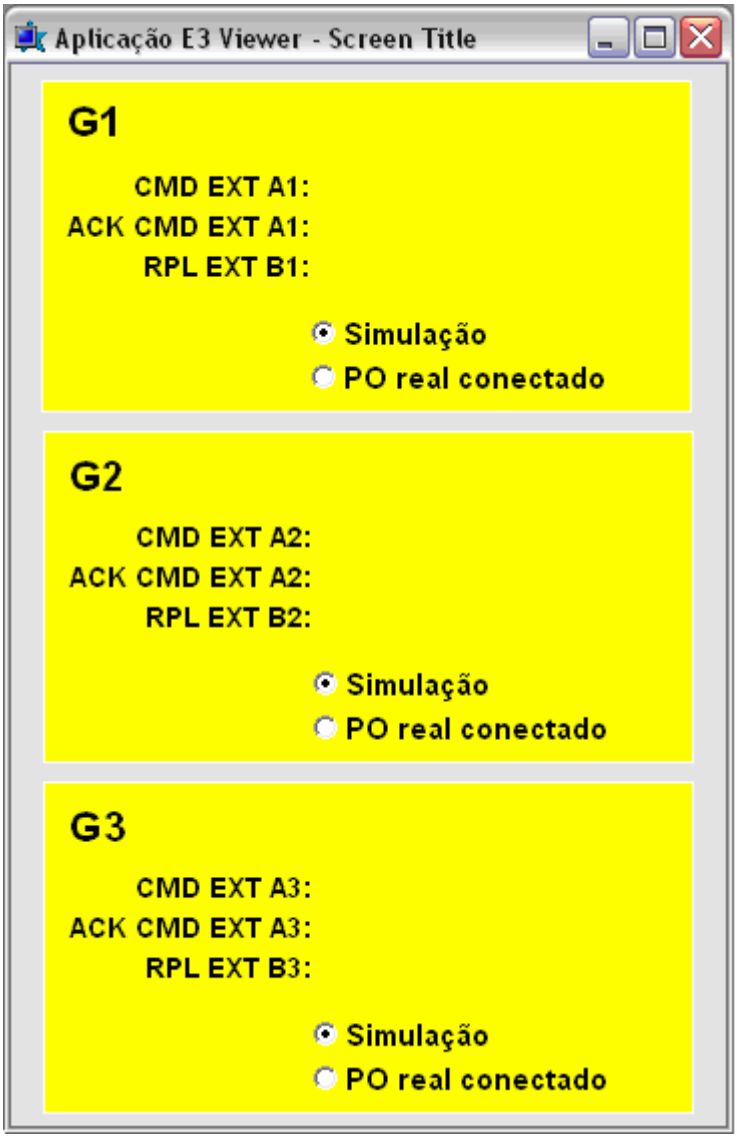


Figura 5.11 – Interface com o usuário para a aplicação do AD para o exemplo de três máquinas

Depois disso, o simulador foi iniciado e foram executados trocas de comandos geradas pelo CLP central. Verificou-se que os comandos gerados no CLP foram corretamente transmitidos para os procedimentos operacionais. Essa validação foi feita contando o número de comandos gerados e o número de execuções dos procedimentos operacionais, conforme pode se observar na Figura 5.12 e Figura 5.13.

	A	B	C	D	E	F	G	H
1					Arena			
2		Real	cmd	rpl	ackcmd	ackrpl	rpl	
3	OS1	0	1	7	1	1	0	
4	OS2	0	1	9	1	1	0	
5	OS3	0	1	4	1	0	7	1

Figura 5.12 – Arquivo de controle para simulação

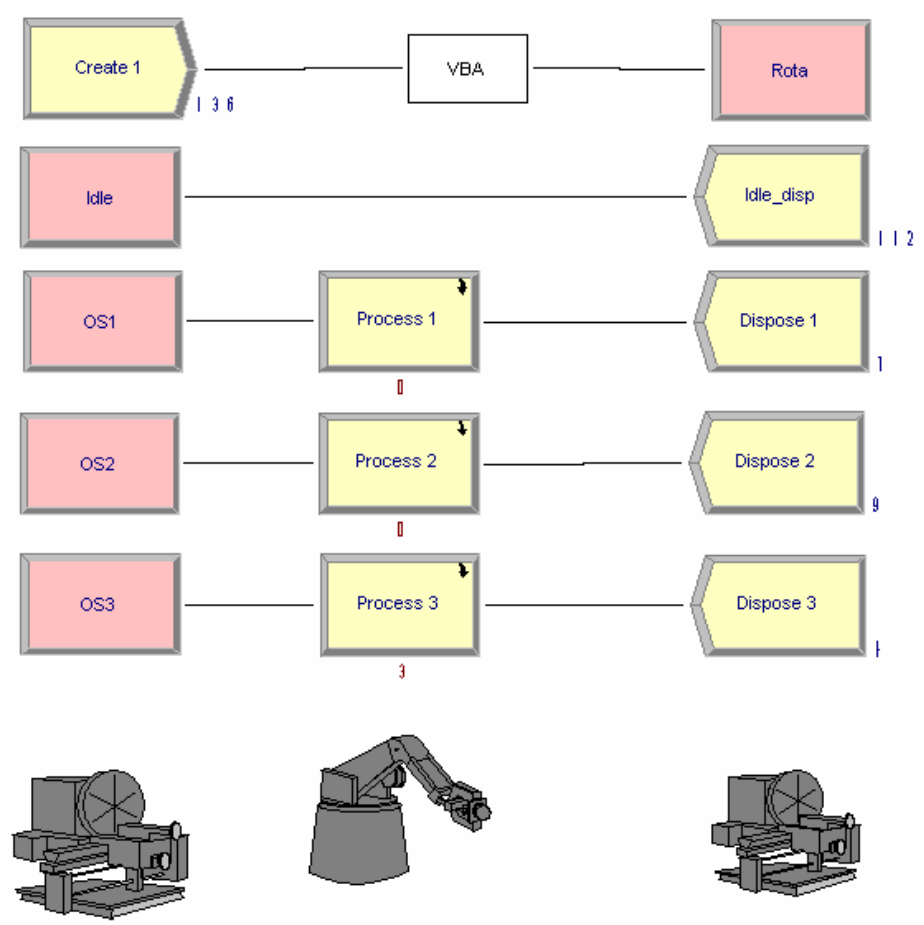


Figura 5.13 – Ambiente de simulação

Desse modo, observa-se que, por meio da comparação da geração de comandos e dos resultados dos procedimentos operacionais simulados, os comandos são interpretados corretamente pelo ambiente de simulação, validando, assim, a etapa de simulação.

Da mesma forma, para que a etapa de Simulação +CTT seja validada, o procedimento operacional 1 é implementado no ambiente real e no ambiente dinâmico, e a variável real é definida como PO real conectada, conforme Figura 5.14.

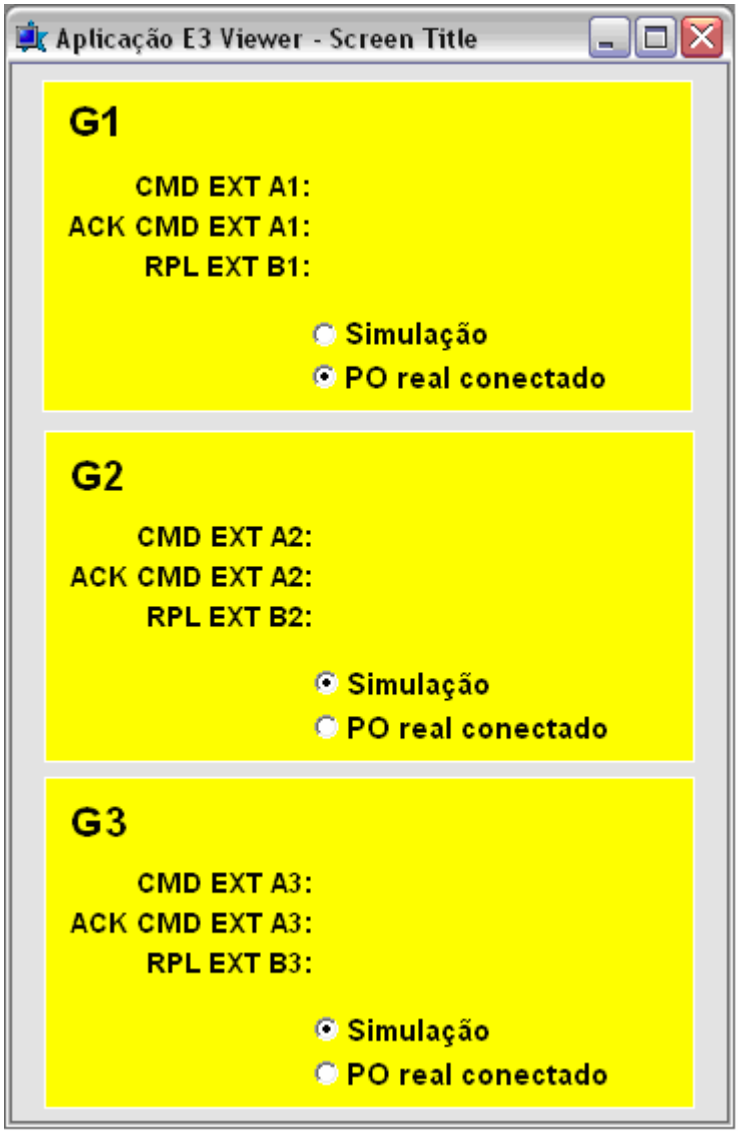


Figura 5.14 – Interface com o usuário para a aplicação do AD para o exemplo de três máquinas, sendo uma implementada no ambiente real.

Assim também se verificou que os comandos referentes aos procedimentos operacionais 2 e 3 foram executados no ambiente de simulação e por ele devolvidos às respectivas respostas e, da forma esperada, os comandos referentes ao procedimento operacional 1 foram executados, mas as respectivas respostas não foram devolvidas, validando então esta etapa.

Finalmente, depois de definir todos os procedimentos operacionais como PO real conectado, observou-se que todos eles foram executados no ambiente real e também no ambiente simulado, mas as respostas não foram devolvidas pelo ambiente real, o que comprova a funcionalidade dessa sistemática.

6 CONCLUSÃO

Em uma análise geral, foi possível evidenciar no capítulo 1, um contexto de trabalhos que buscam soluções de controle para sistemas de manufatura ágil. Hoje, cada vez mais as indústrias necessitam flexibilidade em seus processos de manufatura. Aliada a essa flexibilidade, percebe-se também que a agilidade na reconfiguração das plantas é imprescindível. Outro fator importante é a controlabilidade da planta, o que permite melhores ações de controle sobre ela.

Desta forma, observa-se que a teoria de controle supervisório atende às necessidades de flexibilidade, controlabilidade e reconfiguração em um escopo de manufatura ágil. Nessa linha, a TCS proposta por Ramadge e Wonham (1989) é amplamente trabalhada e aprimorada em trabalhos desenvolvidos por autores, conforme verificado no capítulo 2. Aliado à TCS, o ciclo de desenvolvimento elaborado por Busetti e Santos (2006) fornece sustentação para a implementação de uma metodologia que leva a TCS mais próxima às teorias de manufatura ágeis. Porém, como analisado na revisão bibliográfica, não se encontram em literatura específica esforços no desenvolvimento de uma sistemática que implemente a TCS em ambientes reais; ficam somente expostas as metodologias de tratamentos de problemas e métodos de modelagem.

Conforme exemplificado por Bullock et all (2004), verificou-se que a metodologia HiL apresenta soluções para a implementação de uma sistemática que simula e implementa sistemas de manufatura baseadas na TCS. Porém, como essa metodologia tem estreita relação com o tempo real, implementações usando o HiL impõem uma alta disponibilidade do *hardware* e *software* para atender esse limite de tempo. Desta forma, verificou-se que esses limites são impostos principalmente pela plataforma de *hardware* e *software* na qual a metodologia é implementada.

Desse modo, foi proposta nesse trabalho uma sistemática para complementar a etapa de implementação do ciclo de desenvolvimento de Busetti e Santos (2006) e que contemplasse as três fases desta etapa.

Como a metodologia HiL exige simulação em tempo real para que seja completa, uma adaptação foi exigida para que a simulação pudesse ocorrer em plataformas computacionais mais simples e de fácil acesso. Porém, para garantir o

máximo de desempenho na simulação, uma seqüência de controle de troca de comando e resposta de controles foi desenvolvida. Essa seqüência garante a simulação correta e permite que, utilizando *hardware* e *software* com disponibilidade de tempo real, seja possível uma aproximação da simulação em tempo real.

Também foi possível desenvolver uma distribuição das camadas da estrutura de controle no ambiente de simulação. Com o uso do AD, uma maneira de conectar a planta simulada a subsistemas reais foi implantada de forma gradativa e validou seu funcionamento integrado, parcial ou totalmente, independentemente se entre planta simulada, planta simulada e planta real, ou planta real.

Essa sistemática foi implementada e validada em um sistema de manufatura padrão, composto por 3 equipamentos e dois *buffers*, e, inicialmente, todos os procedimentos operacionais foram somente simulados. Gradativamente, foram inseridos procedimentos operacionais reais em sistemas físicos reais e foram feitas comparações entre eles. Concluiu-se que o comportamento da planta em ambos os casos era o esperado, comprovando-se que a aplicação desta sistemática complementa de forma satisfatória o ciclo de desenvolvimento de Busestti e Santos (2006) e fornece suporte à implantação de sistemas de controle baseados na TCS.

Outro ponto a ser observado é que, nesse caso, detectou-se que a simulação executada com base em tempo real, como proposto pela metodologia HiL, não pode ser usada com recursos computacionais menos sofisticados, como *hardware* do tipo *personal computer* (PC) e sistemas operacionais de alto nível (Microsoft Windows/ Linux). Outro limitante encontrado foi o fato de que o estabelecimento de um canal de comunicação entre o ambiente de simulação e o AD requer o uso de uma terceira tecnologia para padronizar a troca de informações, o que impõe, além de perda de desempenho na simulação, um esforço em dominar o novo ambiente desenvolvido e programar em diversas linguagens de programação, simultaneamente.

Outro fato importante a concluir é que a implementação da estrutura de controle supervisório, usando a simulação como apoio a sua implementação, garante mais segurança e a redução de custos e de tempo. Uma vez que os supervisores são simulados e validados com o ambiente de simulação, é possível garantir com maior precisão que eles funcionarão corretamente quando executados totalmente integrados com a planta física, garantindo assim que o supervisor atue com a segurança necessária sobre a planta. Da mesma forma, uma vez que é

possível simular a implementação dos procedimentos operacionais e sistemas físicos, pode-se estudar alternativas de implementação da planta física de forma a otimizar o uso dos recursos, ganhando em tempo de execução e economia.

Como continuidade a essa trabalho, propõe-se a otimização do uso da plataforma de simulação para garantir uma proximidade maior à simulação em tempo real. Também se propõe a análise de alternativas mais simples na implantação do canal de comunicação entre o ambiente de simulação e o AD, para assim, minimizar as dificuldades encontradas no desenvolvimento deste trabalho.

Como contribuições complementares a esse trabalho foram publicado um artigo sobre o assunto (Diogo et al., 2008).

REFERÊNCIAS

ABDIY, Mohammad Reza; LABIBY, Ashraf W. **A design strategy for reconfigurable manufacturing systems (RMSs).** Using analytical hierarchical process (AHP): a case study. v. 41. Int. J. Prod., 2003.

BERTO, Rosa M.V.S. e NAKANO, Davi N. **Metodologia da pesquisa e a Engenharia de Produção**. Engep, Niterói,1998

BOYD, John; THEYYUNNI, Roger. **Development of a real-time simulation system**. Embedded Systems Programming. ADI. 2006.

BULLOCK, Darcy; JOHNSON, Brian; WELLS, Richard B; KYTE, Michael; LI, Zhen. **Hardware-in-the-Loop simulation**,Transportation Research. Part C 12. 2004 73–89

BUSETTI, Marco Antonio. **Prozeβkopplung mittels einer VMEbus-basierten Hardware-Plattaform für eine verteilte Simulationsumgebung mechatronischer Systeme**. Tese de Doutorado. Alemanha, Universidade de Paderborn, 1998.

BUSSETI, Marco Antonio; SANTOS, Eduardo Alves Portela. **A project methodology applied to automated and integrated manufacturing systems**. Third International Conference on Production Research Americas' Region: Curitiba, Brazil, 2006.

CASSANDRAS, G.C.; LAFORTUNE, S., 1999 **Introduction to Discrete Event Systems**. KLUWER Academic Publishers, Massachusetts, USA.

CRAVOTTA, Robert. **Mixing the Real with the Virtual.** EDN , Waltham, maio de 2005.

CURY, José Eduardo Ribeiro. **Teoria de controle supervisório de sistemas a eventos discretos**. Quinto Simpósio Brasileiro de Automação Inteligente. Canela, Brazil, 2001.

DIOGO, R. A.; VICARI, C. A.; LOURES, E. F. R.; BUSETTI, M. A.; SANTOS, E. A. P. **An Implementation environment for automated manufacturing systems.** 17th IFAC World Congress. Seoul, South Korea, 2008.

FABIAN, Martin; HELLGREN, Anders. **PLC-based implementation of Supervisory Control for Discrete Event Systems**. Proceedings of the 37th IEEE Conference on Decision & Control. Florida, 1998.

GANGULI, A.; DERAEMAEKER, A.; HORODINCA, M.; PREUMONT, A. **Active damping of chatter in machine tools: demonstration with a 'hardware-in-the-loop' simulator**. Proc. IMechE Vol. 219 Part I: J. *Systems and Control Engineering* – pg 359 - 369 , Iasi, Romania, 2005

GERTOSIO, C.; Mebarki, N.; DUSSAUCHOY, A. **Modeling and simulation of the control framework on a flexible manufacturing system**. International Journal of Production Economics. Elsevier. P. 285-293. France, 2000.

HELLGREN, Anders; FABIAN, Martin; LENNARTSON, Bengt. **Modular implementation of Discrete Event Systems as sequential function charts applied to an assembly cell**. Proceedings of the 2001 IEEE International Conference on Control Applications. Mexico City, 2001

HELLGREN, Anders; LENNARTSON, Bengt; FABIAN, Martin. **Modelling and PLC-based implementation of modular supervisory control**. Proceedings of the Sixth International Workshop on Discrete Event Systems (WODES'02). IEEE Computer Society. Göteborg, Sweden, 2002.

HIBINO, H.; INUKAI, T.; FUKUDA, Y. **Efficient manufacturing system implementation based on combination between real and virtual factory**. International Journal of Production Research, 44, 18–19, London, 2006.

HOLST, L., BOLMSJÖ, G.; RANDELL, L.; NORGREN, J. **Integrating simulation into manufacturing system development: A methodological framework**. In: Proceedings of the Twelfth Annual Conference of the Production and Operations Management Society, POM-2001, Orlando, USA. 2001.

K. Iwata, M. Onosato , K. Teramoto and S. Osaki **A modelling and simulation architecture for virtual manufacturing systems**. CIRP Annals. Manufacturing technology. V. 4. Issue 1, 1995.

KOCIJAN, Jus; KARBA, Richard. **A chemical process application of multivariable control hardware and algorithm testing by means of simulation**. Simulation and Pratice Theory. Elsevier. P. 153-165, Slovenia, 1997.

KUMAR, R.; GARG, V. K. **Modeling and Control of Logical Discrete Event Systems**, Kluwer Academic Publishers, 1995.

LAUZON, S.C.; MA, A. K. L.; MILLS, J. K.; BENHABIB, B. **Application of discrete-event-system theory to flexible manufacturin*g***. 1995 IEEE International Conference on Robotics and Automation. Nagoya, Japan, 1995.

LEDUC, R. J. **PLC Implementation of a DES supervisor for a manufacturing testeb**: an implementations perspective. M.A.Sc. Thesis, Dept. of Elect. and Comp. Eng., University of Toronto, Canada, 1996.

LIN, L.; JIANG, Z. **A hybrid supervisory control approach for virtual product systems**. International Journal of Advanced Manufacturing Technologies. Springer, Verlag, London, 2006.

LOURES, Eduardo de Freitas Rocha. **VIEnCoD** – Proposta de um ambiente CACSD baseado em Plataforma de Instrumentação Virtual e Matlab. Dissertação de Mestrado. PUC-PR, Brasil, 1999.

MISSELHORN, W. E.; THERON, N. J.; ELS, P. S. **Investigation of hardware-in-the-Loop for use in suspension development**. Vehicle System Dynamics. Taylor & Francis Group. v. 44. n. 1. p. 65-81. South Africa, 2006.

MOORE, P.R.; PU, J.; NG, H.C.; WONG, C.B.; CHONG, S. K.; CHEN, X.; ADOLFSSON, J.; OLOFSG, P.; LUNDGREN, J.O. **Virtual engineering: an integrated approach to agile manufacturing machinery design and control**. Mechatronics 13, 2003.

NAKANO, Davi N. e FLEURY, Afonso C. C. **Métodos de pesquisa na Engenharia de Produção**. Departamento de Engenharia de Produção. Escola Politécnica, Piracicaba, 1996

PLUMMER, A. R. **Model-in-the-Loop testing**. Proceedings of IMechE. v. 220. Part I. J. Systems and Control Engineering, London, 2006.

PROAKIS, John G.; DIMITRIS, G. Manolakis. **Digital signal processing: principles, algorithms, and applications**. 2. ed. New York: MacMillan, 1992.

QUEIROZ, M.H. de; **Controle Supervisório Modular de Sistemas de Grande Porte**. Florianópolis. Mestrado (Dissertação em Engenharia Elétrica) – Centro Tecnológico,Universidade Federal de Santa Catarina. 2000.

QUEIROZ, M. H.;SANTOS, Eduardo A.P.; CURY, J. E. R. **Síntese modular do controle supervisório em diagrama escada para uma célula de manufatura**. V Simpósio Brasileiro de automação Inteligente, Canela –RS, 2001.

QUEIROZ, Max H. de; CURY, José E. R. **Modular supervisory control of large scale discrete-event systems**. Proceedings of the Sixth International Workshop on Discrete Event Systems (WODES'2000). IEEE Computer Society, 2000.

QUEIROZ, M. H; CURY, J. E. R., 2002. **Controle Supervisório Modular de Sistemas de Manufatura** *Revista Controle & Automação*, Vol. 13, Nº 2, Agosto

QUEIROZ, Max H. de; CURY, José E. R. **Systhesis and implementation of local modular supervisory control for a manufacturing cell**. Proceedings of the Sixth International Workshop on Discrete Event Systems (WODES'02). IEEE Computer Society, 2002.

RAMADGE, Peter J.G.; WONHAM, W. **The control of discrete-event systems**. Proceedins Of THE IEEE, Vol 77, no 1, 1989.

Rockwell *Software*. Disponível em www.rockwellautomation.com/rockwell*software*. Ultima visualização em fevereiro de 2008.

SILVA, Solange da. **Proposta de esquemas baseados em CAC e reserva de recursos para provisão de qos em redes móveis celulares com suporte ao tráfego multimídia**. Universidade Federal de Uberlândia, MG, 2005.

SOFTWARE ELIPSE. Disponível em www.elipse.com.br. Última visualização em fevereiro de 2008.

STOEPPLER, Guido; MENZEL, Thomas; DOUGLAS, Steve. **Hardware-in-the-Loop simulation of machine tools and manufacturing systems**. IEE Computing & Control Engineering. March, 2005.

THIOLLENT, M. **Metodologia da pesquisa-ação**. São Paulo: Cortez, 1994.

VIEIRA, A. D.; CURY, J. E. R.; QUEIROZ, M. H. **Implementação distribuída em controladores lógico programáveis de estrutura de controle supervisório de sistemas a eventos discretos**. Anais do VI Induscon. 2006.

VIEIRA, A. D.; CURY, J. E. R.; QUEIROZ, M. H. **Distribuição de estrutura de controle supervisório de sistemas a eventos discretos.** In. Congresso Brasiliteiro de Automática, 2004. p. 704-709

VIEIRA, A. D. **Modelo de implementação do controle de sistemas a eventos discretos com aplicação da Teoria de Controle Supervisório**. Tese de doutorado. Universidade Federal de Santa Catarina, 2008.

W. E. MISSELHORN; N. J. THERON; P. S. ELS. **Investigation of hardware-in-the-Loop for use in suspension development**. Vehicle System Dynamics. v. 44, n. 1. Janeiro de 2006.

ZHANG, Jie; CHAN, Felix T. S.; LI, Peigen; LAU, Henry C. W.; IP, Ralph W. L.; SAMARANAYAK, P. **Investigation of the reconfigurable control system for an agile manufacturing cell**. Int. j. prod. res., 2002, vol. 40, no. 15, 3709-3723.

APÊNDICE

Apêndice A – Programação Arena VBS

01. Programaçao VBS para implementação dos blocos de funções e fluxo de controle de trocas de comandos e respostas do ambiente de simulação implementado no Arena

```
Option Explicit

Dim g_Model As Arena.Model
Dim g_SIMAN As Arena.SIMAN
Dim g_XLInputFile As Integer
Dim g_XLOutputFile As Integer
Dim g_inputRow As Long
Dim g_outputRow As Long
Dim g_timeIndex As Long
Dim g_partIndex As Long
Dim g_quantityIndex As Long
Dim g_processTimeIndex As Long
Dim g_ArenaDir As String
Dim g_ProcessSchedule As Excel.range
Dim g_ProcessData As Excel.range
Dim disable1 As String
Dim disable2 As String
Dim disable3 As String


Private Sub ModelLogic_RunBeginSimulation()
    smutils_InitializeExcel False, 1
    g_XLInputFile = smutils_OpenExcelWorkbook("c:\final\completo.xls")
    g_XLOutputFile = smutils_OpenExcelWorkbook("c:\final\completo.xls")
    MsgBox "Iniciar Simulação"
End Sub

' Efetuar leitura dos comandos
Private Sub VBA_Block_1_Fire()
    'Definie as variáveis
    Dim s As SIMAN
    Set s = ThisDocument.Model.SIMAN
    Dim com1 As String
    Dim ackreadar1 As String
    Dim com2 As String
```

```
   Dim ackreadar2 As String
   Dim com3 As String
   Dim ackreadar3 As String
   Dim real1 As String
   real1 = smutils_XL.Workbooks(1).Worksheets(1).range("B3").value
   If real1 = "1" Then
   disable1 = "1"
   Else
   disable1 = "0"
   End If
   Dim real2 As String
   real2 = smutils_XL.Workbooks(1).Worksheets(1).range("B4").value
   If real2 = "1" Then
   disable2 = "1"
   Else
   disable2 = "0"
   End If
   Dim real3 As String
   real3 = smutils_XL.Workbooks(1).Worksheets(1).range("B5").value
   If real3 = "1" Then
   disable3 = "1"
   Else
   disable3 = "0"
   End If
   'Inicia processo de leitura dos comando para a máquina 1
   'Verifica se esta 0 ou 1 E3 gravou na celula D3
   g_outputRow = 3
   ackreadar1 = smutils_XL.Workbooks(1).Worksheets(1).range("E3").value
   If ackreadar1 = "0" Then 'Se for 0 o comando ja foi gravado pelo E3
      com1 = smutils_XL.Workbooks(1).Worksheets(1).range("C3").value 'Lê o
comando na C3
      If com1 = "1" Then 'Verifica se o comando foi habilidade, estado 1
         ackreadar1 = 1 'Define 1 para o ACK read do Arena informando que passou
pela leitura
         g_outputRow = 3
         If disable1 = "0" Then
            smutils_WriteExcelValue g_XLOutputFile, "Plan1", "E" & g_outputRow,
ackreadar1 'Grava o ACR read no Excel informando que passou pela Leitura
         End If
         'Como o comando esta habilitado escolhe a rota da máquina 1
         s.VariableArrayValue(s.SymbolNumber("Destin")) = 2
         Exit Sub
```

```
      Else 'Aqui roda se o comando nao foi habilitado , estado 0
         ackreadar1 = 1 'Definne 1 para o ACK read informando que passou pela
leitura
         g_outputRow = 3
         If disable1 = "0" Then
            smutils_WriteExcelValue g_XLOutputFile, "Plan1", "E" & g_outputRow,
ackreadar1 'Grava o ACR read no Excel informando que passou pela Leitura
         End If
         'Próximo passo manda a entidade para o estado de idle
         s.VariableArrayValue(s.SymbolNumber("Destin")) = 1
         'Exit Sub
      End If
   Else 'Se o valor for 1, significa que o E3 nao gravou um comando novo
      com1 = smutils_XL.Workbooks(1).Worksheets(1).range("C3").value 'Lê o
comando na B3
      If com1 = "0" Then
         'ackreadar1 = 0 'Define 0 para o ACK read do Arena informando que passou
pela leitura
         'g_outputRow = 3
         'smutils_WriteExcelValue g_XLOutputFile, "Plan1", "E" & g_outputRow,
ackreadar1
         's.VariableArrayValue(s.SymbolNumber("Destin")) = 1
      End If
      s.VariableArrayValue(s.SymbolNumber("Destin")) = 1
   End If
   'Inicia processo de leitura dos comando para a máquina 2
   g_outputRow = 4
   'Verifica se esta 0 ou 1 . E3 gravou na celula D4
   ackreadar2 = smutils_XL.Workbooks(1).Worksheets(1).range("E4").value
   If ackreadar2 = "0" Then 'Se for 0 o comando ja foi gravado pelo E3
      com2 = smutils_XL.Workbooks(1).Worksheets(1).range("C4").value 'Lê o
comando na B4
      If com2 = "1" Then 'Verifica se o comando foi habilidade, estado 1
         ackreadar2 = 1 'Define 1 para o ACK read do Arena informando que passou
pela leitura
         If disable2 = "0" Then
            smutils_WriteExcelValue g_XLOutputFile, "Plan1", "E" & g_outputRow,
ackreadar2 'Grava o ACR read no Excel informando que passou pela Leitura
         End If
         'Como o comando esta habilitado escolhe a rota da máquina 1
         s.VariableArrayValue(s.SymbolNumber("Destin")) = 3
         Exit Sub
```

```
        Else 'Aqui roda se o comando nao foi habilitado , estado 0
            ackreadar2 = 1 'Definne 1 para o ACK read informando que passou pela
leitura
            If disable2 = "0" Then
            smutils_WriteExcelValue g_XLOutputFile, "Plan1", "E" & g_outputRow,
ackreadar2 'Grava o ACR read no Excel informando que passou pela Leitura
            End If
            'Próximo passo manda a entidade para o estado de idle
            s.VariableArrayValue(s.SymbolNumber("Destin")) = 1
            'Exit Sub
        End If
    Else 'Se o valor for 1, significa que o E3 nao gravou um comando novo
        com2 = smutils_XL.Workbooks(1).Worksheets(1).range("C4").value 'Lê o
comando na B4
        If com2 = "0" Then
            'ackreadar2 = 0 'Define 0 para o ACK read do Arena informando que passou
pela leitura
            'smutils_WriteExcelValue g_XLOutputFile, "Plan1", "E" & g_outputRow,
ackreadar2
            's.VariableArrayValue(s.SymbolNumber("Destin")) = 1
        End If
        s.VariableArrayValue(s.SymbolNumber("Destin")) = 1
    End If
    'Inicia processo de leitura dos comando para a máquina 3
    g_outputRow = 5
    'Verifica se esta 0 ou 1 . E3 gravou na celula D5
    ackreadar3 = smutils_XL.Workbooks(1).Worksheets(1).range("E5").value
    If ackreadar3 = "0" Then 'Se for 0 o comando ja foi gravado pelo E3
        com3 = smutils_XL.Workbooks(1).Worksheets(1).range("C5").value 'Lê o
comando na B5
        If com3 = "1" Then 'Verifica se o comando foi habilidade, estado 1
            ackreadar3 = 1 'Define 1 para o ACK read do Arena informando que passou
pela leitura
            If disable3 = "0" Then
            smutils_WriteExcelValue g_XLOutputFile, "Plan1", "E" & g_outputRow,
ackreadar3 'Grava o ACR read no Excel informando que passou pela Leitura
            'Como o comando esta habilitado escolhe a rota da máquina 1
            End If
            s.VariableArrayValue(s.SymbolNumber("Destin")) = 4
            Exit Sub
        Else 'Aqui roda se o comando nao foi habilitado , estado 0
```

```
            ackreadar3 = 1 'Definne 1 para o ACK read informando que passou pela
leitura
            If disable3 = "0" Then
            smutils_WriteExcelValue g_XLOutputFile, "Plan1", "E" & g_outputRow,
ackreadar3 'Grava o ACR read no Excel informando que passou pela Leitura
            End If
            'Próximo passo manda a entidade para o estado de idle
            s.VariableArrayValue(s.SymbolNumber("Destin")) = 1
            'Exit Sub
        End If
    Else 'Se o valor for 1, significa que o E3 nao gravou um comando novo
        com3 = smutils_XL.Workbooks(1).Worksheets(1).range("C5").value 'Lê o
comando na B4
        If com3 = "0" Then
            'ackreadar3 = 0 'Define 0 para o ACK read do Arena informando que passou
pela leitura
            'smutils_WriteExcelValue g_XLOutputFile, "Plan1", "E" & g_outputRow,
ackreadar3
            's.VariableArrayValue(s.SymbolNumber("Destin")) = 1
        End If
        s.VariableArrayValue(s.SymbolNumber("Destin")) = 1
    End If
End Sub
Private Sub ModelLogic_RunEndSimulation()
    'smutils_SaveExcelWorkbook g_XLOutputFile, "c:\final\completo.xls"
    If Not smutils_XLInitialized Then Exit Sub
    smutils_XL.DisplayAlerts = False
    'smutils_XL.Quit
    Set smutils_XL = Nothing
    smutils_XLInitialized = False
    Set g_ProcessSchedule = Nothing
    Set g_ProcessData = Nothing
End Sub

Private Sub VBA_Block_11_Fire()
    Dim real1 As String
    real1 = smutils_XL.Workbooks(1).Worksheets(1).range("B3").value
    If real1 = "1" Then
    disable1 = "1"
    Else
    disable1 = "0"
    End If
```

```
End Sub


Private Sub VBA_Block_12_Fire()
   Dim real2 As String
   real2 = smutils_XL.Workbooks(1).Worksheets(1).range("B4").value
   If real2 = "1" Then
   disable2 = "1"
   Else
   disable2 = "0"
   End If
End Sub


Private Sub VBA_Block_13_Fire()
   Dim real3 As String
   real3 = smutils_XL.Workbooks(1).Worksheets(1).range("B5").value
   If real3 = "1" Then
   disable3 = "1"
   Else
   disable3 = "0"
   End If
End Sub

Private Sub VBA_Block_7_Fire()
   Dim resposta1 As Integer
   Dim acrwrite1 As Integer
   resposta1 = smutils_XL.Workbooks(1).Worksheets(1).range("D3").value
   resposta1 = resposta1 + 1
   acrwrite1 = 1
   g_outputRow = 3
   If disable1 = "0" Then
   smutils_WriteExcelValue g_XLOutputFile, "Plan1", "D" & g_outputRow, resposta1
   smutils_WriteExcelValue g_XLOutputFile, "Plan1", "F" & g_outputRow, acrwrite1
   End If
End Sub

Private Sub VBA_Block_8_Fire()
   Dim resposta2 As Integer
   Dim acrwrite2 As Integer
   resposta2 = smutils_XL.Workbooks(1).Worksheets(1).range("D4").value
   resposta2 = resposta2 + 1
```

```
    acrwrite2 = 1
    g_outputRow = 4
    If disable2 = "0" Then
    smutils_WriteExcelValue g_XLOutputFile, "Plan1", "D" & g_outputRow, resposta2
    smutils_WriteExcelValue g_XLOutputFile, "Plan1", "F" & g_outputRow, acrwrite2
    End If
End Sub


Private Sub VBA_Block_9_Fire()
    Dim resposta3 As Integer
    Dim acrwrite3 As Integer
    resposta3 = smutils_XL.Workbooks(1).Worksheets(1).range("G5").value
    resposta3 = resposta3 + 1
    acrwrite3 = 1
    g_outputRow = 5
    If disable3 = "0" Then
    smutils_WriteExcelValue g_XLOutputFile, "Plan1", "G" & g_outputRow, resposta3
    smutils_WriteExcelValue g_XLOutputFile, "Plan1", "H" & g_outputRow, acrwrite3
    End If
End Sub

Private Sub VBA_Block_10_Fire()
    Dim resposta4 As Integer
    Dim acrwrite4 As Integer
    resposta4 = smutils_XL.Workbooks(1).Worksheets(1).range("D5").value
    resposta4 = resposta4 + 1
    acrwrite4 = 1
    g_outputRow = 5
    If disable3 = "0" Then
    smutils_WriteExcelValue g_XLOutputFile, "Plan1", "D" & g_outputRow, resposta4
    'smutils_WriteExcelValue g_XLOutputFile, "Plan1", "E" & g_outputRow, acrwrite4
    End If
End Sub
```

2) Funções definidas para controle de acesso ao Microsoft Excel usadas no Arena

```
'=============================================================================
'
' This code module contains utility functions that you can use to access Microsoft Excel worksheet
'  data easily. If you'd like to use it in other models/programs, export it to a .bas file and
'  add that .bas file as a code module in your other projects.
'
'
'=============================================================================
' Microsoft Excel Interface Functions ... VBA/VB
'   smutils_InitializeExcel( visible As Boolean, errorHandling As Integer )
'   smutils_OpenExcelWorkbook( filename As String ) As Integer
'   smutils_NewExcelWorkbook() As Integer
'   smutils_SaveExcelWorkbook( fileHandle As Integer, filename As String )
'   smutils_ReadExcelValue( fileHandle As Integer, sheetName As String, dataRange As String ) As Variant
'   smutils_ReadExcelRange( fileHandle As Integer, sheetName As String, rangeName As String ) As Excel.Range
'   smutils_WriteExcelValue( fileHandle As Integer, sheet As String, dataRange As String, value As Variant )
'   smutils_ExitExcel()
'
'=============================================================================

Option Explicit

'
'=============================================================================
' Global Variables for Excel Functions
Public smutils_XL As Excel.Application
Public smutils_XLInitialized As Boolean
Public smutils_XLErrorHandling As Integer

' errorHandling constants
Const smutils_ErrorsIgnore = 0
Const smutils_ErrorsDisplayMessage = 1
```

```
Const smutils_ErrorsLogToFile = 2
'
================================================================
============================

Public Sub smutils_InitializeExcel(ByVal visible As Boolean, ByVal errorHandling As
Integer)
    If smutils_XLInitialized Then Exit Sub

    On Error Resume Next
    Set smutils_XL = GetObject(, "Excel.Application")
    If Err.Number <> 0 Then
        On Error GoTo InitExcelError
        Set smutils_XL = CreateObject("Excel.Application")
    End If
    Err.Clear

    smutils_XLInitialized = True
    smutils_XL.visible = True
    smutils_XLErrorHandling = errorHandling
    If visible Then smutils_XL.visible = True
    Exit Sub

' ===== Errors =====
InitExcelError:
   smutils_HandleXLError ("Error " & Err & " initializing Excel.")
   Exit Sub
End Sub
Public Function smutils_OpenExcelWorkbook(filename As String) As Integer
    smutils_OpenExcelWorkbook = -1
    On Error GoTo InitializeError
    If Not smutils_XLInitialized Then
        smutils_InitializeExcel False, smutils_ErrorsDisplayMessage
    End If
    On Error GoTo OpenError
    With smutils_XL
        .Workbooks.Open (filename)
        smutils_OpenExcelWorkbook = .Workbooks.Count
    End With
    Exit Function

' ===== Errors =====
```

```
InitializeError:
    smutils_HandleXLError ("Error " & Err & " initializing Excel.")
    Exit Function
OpenError:
    If Err = 1004 Then
        smutils_HandleXLError ("Cannot find file " & filename & ".")
    Else
        smutils_HandleXLError ("Error " & Err & " opening file " & filename & ".")
    End If
    Exit Function
End Function
Public Function smutils_NewExcelWorkbook() As Integer
    smutils_NewExcelWorkbook = -1
    On Error GoTo InitializeError
    If Not smutils_XLInitialized Then
        smutils_InitializeExcel False, smutils_ErrorsDisplayMessage
    End If
    On Error GoTo AddError
    With smutils_XL
        .Workbooks.Add
        smutils_NewExcelWorkbook = .Workbooks.Count
    End With
    Exit Function

' ===== Errors =====
InitializeError:
    smutils_HandleXLError ("Error " & Err & " initializing Excel.")
    Exit Function
AddError:
    smutils_HandleXLError ("Error " & Err & " adding a workbook.")
    Exit Function
End Function
Public Sub smutils_SaveExcelWorkbook(ByVal fileHandle As Integer, ByVal filename
As String)
    If Not smutils_XLInitialized Then GoTo ExcelNotInitializedError
  On Error Resume Next
    With smutils_XL
        If fileHandle < 1 Or fileHandle > .Workbooks.Count Then
            smutils_HandleXLError ("Error: Trying to save workbook with invalid file
handle.")
            Exit Sub
        End If
```

```
        .Workbooks(fileHandle).SaveAs filename
        .Workbooks(fileHandle).Close saveChanges:=False
    End With
    Exit Sub

' ===== Errors =====
ExcelNotInitializedError:
    MsgBox "Error writing to Excel file. Excel first must be initialized" & Chr(13) & _
        "and a workbook must be opened. Use NewExcelWorkbook or
OpenExcelWorkbook."
    Exit Sub
End Sub
Public Function smutils_ReadExcelValue(ByVal fileHandle As Integer, ByVal
sheetName As String, ByVal dataRange As String) As Variant
    On Error GoTo ReadError
    If Not smutils_XLInitialized Then GoTo ExcelNotInitializedError
    With smutils_XL
        If fileHandle < 1 Or fileHandle > .Workbooks.Count Then
            smutils_HandleXLError ("Error: Trying to read from unopened spreadsheet.")
            Exit Function
        End If
        smutils_ReadExcelValue =
.Workbooks(fileHandle).Worksheets(sheetName).range(dataRange).value
    End With
    Exit Function

' ===== Errors =====
ExcelNotInitializedError:
    MsgBox "Error reading from Excel file. Excel first must be initialized" & Chr(13) & _
        "and a workbook must be opened. Use NewExcelWorkbook or
OpenExcelWorkbook."
    Exit Function
ReadError:
    smutils_HandleXLError ("Error " & Err & " reading from file " &
smutils_XL.Workbooks(fileHandle).Name & ".")
    Exit Function
End Function
Public Function smutils_ReadExcelRange(ByVal fileHandle As Integer, ByVal
sheetName As String, ByVal rangeName As String) As Excel.range
    On Error GoTo ReadError
    If Not smutils_XLInitialized Then GoTo ExcelNotInitializedError
    With smutils_XL
```

```
        If fileHandle < 1 Or fileHandle > .Workbooks.Count Then
            smutils_HandleXLError ("Error: Trying to read from unopened spreadsheet.")
            Exit Function
        End If
        Set smutils_ReadExcelRange =
.Workbooks(fileHandle).Worksheets(sheetName).range(rangeName)
    End With
    Exit Function

' ===== Errors =====
ExcelNotInitializedError:
    MsgBox "Error reading from Excel file. Excel first must be initialized" & Chr(13) & _
        "and a workbook must be opened. Use NewExcelWorkbook or
OpenExcelWorkbook."
    Exit Function
ReadError:
    smutils_HandleXLError ("Error " & Err & " reading from file " &
smutils_XL.Workbooks(fileHandle).Name & ".")
    Exit Function
End Function
Public Sub smutils_WriteExcelValue(ByVal fileHandle As Integer, ByVal sheet As
String, _
    ByVal dataRange As String, ByVal value As Variant)
    If Not smutils_XLInitialized Then GoTo ExcelNotInitializedError
    On Error GoTo WriteError
    With smutils_XL
        If fileHandle < 1 Or fileHandle > .Workbooks.Count Then
            smutils_HandleXLError ("Error: Trying to write data to non-initialized
workbook." & Chr(13) & _
                "Use OpenExcelWorkbook or NewExcelWorkbook.")
            Exit Sub
        End If
        .Workbooks(fileHandle).Worksheets(sheet).range(dataRange).value = value
    End With
    Exit Sub

' ===== Errors =====
ExcelNotInitializedError:
    MsgBox "Error writing to Excel file. Excel first must be initialized" & Chr(13) & _
        "and a workbook must be opened. Use NewExcelWorkbook or
OpenExcelWorkbook."
    Exit Sub
```

```
WriteError:
   smutils_HandleXLError ("Error " & Err & " writing to file " &
smutils_XL.Workbooks(fileHandle).Name & ".")
   Exit Sub
End Sub
Private Sub smutils_HandleXLError(ByVal errorString As String)
   If smutils_XLErrorHandling = smutils_ErrorsIgnore Then Exit Sub
   Static errorFile As String, errorFileOpened As Boolean
   Select Case smutils_XLErrorHandling
      Case smutils_ErrorsDisplayMessage
         MsgBox errorString
         Exit Sub
   End Select
End Sub
```

Livros Grátis

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

Milhares de Livros para Download:

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

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