RICARDO ALEXANDRE DIOGO

Um Ambiente de Suporte à Implementação da Arquitetura de Controle Supervisório

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

CURITIBA
2008

RICARDO ALEXANDRE DIOGO

Um Ambiente de Suporte à Implementação da Arquitetura de Controle Supervisório

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia da Produção e Sistemas da 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 Antonio Busetti de Paula.

CURITIBA
2008

“Plante seu jardim e decore sua alma, ao invés de esperar que alguém lhe traga flores.”

William Shakespeare

AGRADECIMENTOS

Este trabalho é dedicado a todos aqueles que me incentivaram ou ajudaram, de forma direta ou indireta, na execução deste projeto.

Agradecimentos especiais são devidos ao Prof. Dr. Eduardo Alves Portela Santos, meu orientador, e ao Prof. Dr. Marco Antonio Busetti de Paula, co-orientador! Agradeço a confiança, revelada pelo convite à realização do programa de mestrado!

Ao Prof. Dr. Agnelo D. Vieira, ao Prof. Dr. Eduardo de Freitas Rocha Loures e ao Prof. Msc. Fernando Deschamps, pela ajuda na resolução de problemas!

Agradeço a todos os professores, também, pela viabilização de oportunidades importantes para a minha vida!

Aos funcionários do Grupo Produtrônica, por me haverem disponibilizado, sempre que possível, as soluções administrativas e a operacionalização dos laboratórios!

À CAPES, pelo apoio financeiro!

À Elipse Software, pelo empréstimo das licenças do *software* Elipse E3 e pelo suporte técnico durante todo o desenvolvimento do projeto! Também ao suporte técnico da Paragon, pelo apoio no desenvolvimento de modelos no *software* de simulação Arena!

A minha família e aos meus amigos, companheiros de todas as horas, até mesmo quando não estive presente! Agradeço por entenderem minha ausência em tantas oportunidades!

A minha amada namorada, Rossana Calegari dos Santos, que me incentivou fortemente no período do curso e em outras decisões ao longo de nossa vida!

A minha mãe, que me deu suporte nas horas de estresse, por haver-me acudido nas noites de frio, por haver-me transformado na pessoa que sou e por estar sempre atenta ao que o filho faz!

E a Deus, porque, de uma maneira ou de outra, Ele pôs Seu dedo neste projeto e me deu condições de executá-lo!

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS

ACS – Arquitetura de Controle Supervisório

AD – Ambiente Dinâmico

AML – Abordagem Modular Local

BD – Base de Dados

BMC – Biblioteca de Modelos de Comunicação

CID – *Controller Interface Device*

CLP – Controlador Lógico Programável

CNC – Comando Numérico Computadorizado

DDE – *Dynamic Data Exchange*

DE – *Dynamic Environment*

DES – *Discrete Event System*

E/S – Entrada/Saída

FB – *Function Block*

FMS – *Flexible Manufacturing System*

HiL – *Hardware-in-the-Loop*

iE – *Industrial Ethernet*

IEC – *International Electrotechnical Comission*

ISO – *International Organization for Standardization*

I/O – *Input/Output*

LMA – *Local Modular Approach*

PCS – Problema de Controle Supervisório

PO – Procedimentos Operacionais

POR – Procedimento Operacional Real

POV – Procedimento Operacional Virtual

RSP – Representação por Sistema Produto

SCADA – *Supervisory Control And Data Acquisition*

SRC – Sistema Real a Controlar

SED – Sistemas a Eventos Discretos

SFC – *Sequential Function Chart*

SM – Supervisores Modulares

SP – Sistema Produto

SCT – *Supervisory Control Theory*

SSC – Sistema a Ser Controlado

SVC – Sistema Virtual a Controlar

TCC – Tecnologias de Comunicação e Controle

TCS – Teoria de Controle Supervisório

TMA – Tecnologias de Manufatura Avançadas

VBA – *Visual Basic for Applications*

RESUMO

O crescimento da competição global tem forçado as empresas a reduzirem cada vez mais o tempo de lançamento dos seus produtos. Como consequência, existe um esforço, tanto por parte das empresas como da academia, no sentido de proverem técnicas e ferramentas para a rápida reconfiguração da arquitetura de controle em sistemas de manufatura. Um sistema de manufatura é um exemplo clássico de um Sistema a Eventos Discretos (SED), isto porque sua dinâmica evolui de acordo com a ocorrência de tais eventos 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 tais problemas. Porém, existem ainda limitações a este formalismo, como a centralização do agente supervisor. Extensões da TCS, como a Abordagem Modular Local (AML), são alternativas para a descentralização do agente supervisor e para se propiciar a reconfiguração de sistemas de manufatura de forma mais rápida. Verificou-se, então, a necessidade de um Ciclo de Desenvolvimento, cuja primeira etapa é a modelagem de SED; a segunda, a síntese dos supervisores modulares, e, finalmente, uma terceira etapa, a implementação. Nesta, o modelo teórico é traduzido numa linguagem compatível com uma plataforma industrial. Este procedimento necessita de métodos de validação antes de o equipamento ser conectado ao sistema de manufatura. Um método de validação possível é a simulação da arquitetura de controle e sua inserção gradativa ao sistema físico a controlar. Este trabalho propõe um Ambiente Dinâmico (AD) de suporte à implementação de uma arquitetura de controle. O objetivo de tal ambiente é validar a arquitetura de controle obtida através da simulação e, gradativamente, permitir a inserção dos controladores reais da planta física. Ao longo da validação, os subsistemas físicos anteriormente simulados são inseridos ao sistema de manufatura, numa técnica conhecida como *Hardware-in-the-Loop* (HiL). Um exemplo de sistema de manufatura é proposto, e, posteriormente, modificado, com a inserção de uma nova máquina, para demonstrar a reconfiguração do sistema. Os resultados obtidos demonstram que a reconfiguração do sistema ocorre sistemática e rapidamente.

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

ABSTRACT

The increasing of global competition has been forcing the companies to reduce more and more the launching time of their products. As a result, companies and academy make efforts to provide techniques and tools for the fast reconfiguration of control systems in manufacturing systems. A manufacturing system is a classic example of a Discrete Event System (DES), because its dynamic evolves according to the occurrence of such events along time. In this context, formal methods for DES modeling, analyses and synthesis, as the Supervisory Control Theory (SCT), are contributing to solve problems. However, there are limitations to the formalism, such as the centralization of the supervisor agent. SCT extensions as the Local Modular Approach (LMA) are alternatives to decentralize the supervisor agent and to provide the reconfiguration of the manufacturing system as fast as possible. There is the need of a development cycle with three stages, where the first is the DES modeling, the second is the modular supervisors' synthesis and finally the third stage is the implementation. In the last stage, the theoretical model is translated to a compatible language with industrial platforms. This procedure needs the validation methods before the equipment is connected to the manufacturing system. A possible validation method is the simulation of the control structure and its gradual insertion to the real system. This work proposes a Dynamic Environment (DE) that gives support to the implementation of a control system. The objective of DE is to validate the control structure obtained through the simulation and allows the gradual insertion of the physical plant's real controllers. Along the validation, the physical subsystems simulated beforehand are inserted into the manufacturing system with the Hardware-in-the-Loop (HiL) technique. An example of manufacturing system is proposed and modified with the insertion of a new machine to demonstrate the system reconfiguration. The results show that the reconfiguration of the system occurs systematically and rapidly.

Keywords: Manufacturing systems, Discrete Event Systems, Supervisory Control Theory, Local Modular Approach, Programmable Logic Controller, Systems Integration, Simulation, Hardware-in-the-Loop, SCADA.

1 INTRODUÇÃO

A Teoria de Controle Supervisório (TCS) foi desenvolvida com o objetivo de prover uma metodologia formal para a síntese de controladores para Sistemas a Eventos Discretos (Ramadge e Wonham, 1989). Em relação à implementação da TCS, muitas abordagens são encontradas na literatura, no sentido de sistematizarem e também difundirem a aplicação da teoria (Fabian e Hellgren, 1998; Hellgren *et al*, 2002; Leduc, 1996; Queiroz e Cury, 2002; Vieira *et al*, 2006; Vieira, 2007).

Porém, uma das dificuldades de aplicação da TCS está relacionada com a implementação de seus resultados. Em linhas gerais, existe necessidade de uma sistemática que traduza o modelo teórico da TCS numa linguagem própria de controladores industriais. O maior desafio, nesse sentido, é fazer com que o código implementado em tais controladores execute adequadamente a mesma ação de controle estabelecida pela TCS.

Outra importante questão relacionada à implementação da TCS é a distribuição da arquitetura de controle. De acordo com Vieira e Cury (2004), a distribuição é fundamental, quando se consideram fatores como a limitação das plataformas utilizadas (número de entradas e saídas, recursos internos) e a modularidade do sistema físico a controlar (geralmente formado por células de trabalhos com respectivos controladores locais). Também é importante citar que a distribuição do sistema físico a controlar permite uma melhor organização do comissionamento da planta, alterações futuras na instalação, manutenção, ou, mesmo, a operação parcial do sistema em caso de falha de um determinado módulo.

Existem trabalhos que estudam a distribuição do sistema físico a controlar, sem utilizar a TCS, como o de Gertosio *et al* (2000), no qual os autores descentralizam a estação de controle e vinculam um supervisor para cada célula de manufatura. Assim, quando uma célula deixa de ser usada ou uma nova é inserida, somente o supervisor vinculado é excluído ou implementado, respectivamente. Porém, esses supervisores só trocam dados com o gerenciador da produção e não com os demais supervisores. Ou seja, a estrutura de controle é hierárquica. Outro problema é que o gerenciador da produção continua centralizado. Este tem a função

de informar aos supervisores sobre o inicio de trabalho de cada célula. Como o gerenciador é monolítico, toda vez que uma mudança é feita, um novo *software* de gerenciamento deve ser implementado no sistema. Neste caso, a modularidade não é observada.

Tratando-se de modularidade, foram desenvolvidos trabalhos que focam a solução de problemas quando utilizam a TCS como ferramenta de programação para Controladores Lógico Programáveis (CLP). Dentre os modelos de implementação propostos, destacam-se os de Queiroz e Cury (2002), Vieira *et al* (2006) e Vieira (2007). Porém, esses trabalhos não tratam da questão sobre a sistematização de implementação gradativa. Questões como testes, ajustes, detecção de erros, validação dos programas e dos modelos não são abordadas.

A implementação gradativa se dá por um conjunto de passos que consistem em construir modelos de subsistemas virtuais, os quais foram determinados de acordo com os subsistemas físicos. Após a modelagem, é possível testá-los através de simulação, analisando-se, assim, a dinâmica de funcionamento virtual. Isto objetiva a correção de erros antes que ocorra a implementação dos subsistemas físicos. Este processo é, então, repetido para todos os outros subsistemas integrantes do sistema de manufatura, até que este esteja, fisicamente, implementado por completo. Estes passos são integrantes do conceito *Hardware-in-the-Loop* (HiL). Este conceito vem sendo muito usado pela indústria aeronáutica, porém, com passos mais lentos na indústria de manufatura. O HiL tem o objetivo de diminuir o tempo de implementação física de um sistema, além de se evitar perdas na construção de protótipos ou, até mesmo, acidentes com operadores por erros de projeto que poderiam ser evitados através da simulação a *priori*.

Para Boyd e Theyyunni (1999) e Stoeppler *et al* (2005), o teste através de simulação é uma maneira rápida de prototipagem e deve ser aplicado a sistemas físicos em tempo real, pois permite a otimização da planta de manufatura. Os autores colocam em suas publicações razões para o uso de HiL em sistemas de simulação em tempo real:

i) pode não haver maneiras viáveis para construção de protótipos físicos;
ii) o controle do ambiente de teste real pode ser muito difícil;
iii) questões de segurança podem tornar os testes de execução impossíveis no sistema;

iv) considerações econômicas ditam o uso da simulação;

v) o tempo de *set-up* é reduzido.

O custo para utilização de HiL pode ser elevado em termos de *software* e em termos de *hardware*, ainda mais quando se deseja que a simulação ocorra em tempo real, pois maiores investimentos devem ser aplicados em tais equipamentos e ferramentas. Porém, torna-se justificável devido à capacidade de se evitarem acidentes e a destruição de protótipos. E, como a cada dia a informática obtém melhor desempenho a menores preços, a simulação acaba se tornando viável (Cravotta, 2005).

Várias ferramentas computacionais de simulação e supervisão são utilizadas no dia-a-dia dos projetistas. Dentre elas, podem ser citadas o Matlab-Simulink (Mathworks, 2008), LabVIEW (National Instruments, 2008), CATIA (Dessault Systemes, 2008) e Unigraphix (Nx3, 2008). *Softwares* como o CATIA, que surgiu no projeto e construção do Boeing 777 (The Boeing Company, 2008; Discovery Channel, 2008) são usados para simulações mecânicas e de usinagem. Ainda podem ser citados o Arena (Rockwell Automation, 2008) e eM-Plant (Siemens, 2008), que são conhecidos por sua capacidade de geração de relatórios em simulações para determinação de processos produtivos; no entanto, também são capazes de trabalhar paralelamente com o sistema de manufatura, através da inserção de *drivers* para comunicação com os controladores da planta. Outras plataformas computacionais como o Elipse SCADA, Elipse E3 (Elipse Software, 2008) e outros, conhecidos como *softwares* supervisórios, têm a finalidade de gerenciamento dos sistemas físicos do chão-de-fábrica.

O uso dessas ferramentas computacionais tem a finalidade de tornar o processo de projeto mais ágil e flexível, porém, flexibilidade e agilidade não estão relacionadas somente ao uso de *softwares*. A manufatura ágil está relacionada com a flexibilidade, e esta, por sua vez, com o tempo de implementação de sistemas de controle. A manufatura ágil foi introduzida no início dos anos 90 como resposta a conseqüentes e rápidas mudanças da nova economia, e é a base para a competitividade em uma economia globalizada (Busetti e Santos, 2006). Não basta ser flexível, um Sistema Flexível de Manufatura (FMS – *Flexible Manufacturing System*) deve ser ágil em seu processo produtivo, tanto para se obter um menor *lead-time* como para realizar a re-configuração do sistema, quando necessário. Uma

re-configuração ágil deve aproveitar o sistema existente, re-configurar os processos ou incluir/excluir outros.

A flexibilidade de sistemas impõe que o tempo de implementação seja o mais curto possível, pois um atraso em relação aos concorrentes pode resultar em perdas incontáveis (Babic, 1999). As demandas de usuários e mercados exigem que os sistemas de manufatura de hoje sejam ágeis e flexíveis; portanto, o correspondente sistema de controle precisa, consequentemente, ser re-utilizável, adaptável e re-configurável (Cai *et al*, 2003), tornando flexíveis tanto o *hardware* como o *software*. Isto promove a competitividade e inovação.

Após várias pesquisas, Wadhwa *et al* (2005) chegaram à conclusão de que a flexibilidade de um FMS habilita de maneira eficaz o gerenciamento das mudanças. Tais mudanças se resumem a como gerenciar os sistemas concorrentes do processo de manufatura. A concorrência de um sistema é definida quando dois ou mais processos precisam ser realizados por uma mesma máquina. Basicamente, isto é ilustrado na Figura 1, na qual se verifica que, quanto maior for a inflexibilidade do sistema, menor será sua concorrência, assim como a variedade de produtos, ou seja, o desequilíbrio será menor. Observa-se, ainda, que, se diminuir a inflexibilidade, menor será o *lead-time,* e, um ponto particularmente importante para este trabalho, menos complexo será o sistema de controle.

Porém, se uma empresa ficar presa à inflexibilidade de seu sistema de manufatura, isso significa que a variedade de seus produtos é muito baixa. Isto faz com que a empresa dependa do consumo de tal pequena gama de produtos, ficando vulnerável à não-aceitação por parte do mercado. Se isto ocorrer, medidas drásticas para manter a empresa viva no mercado devem ser tomadas, como a

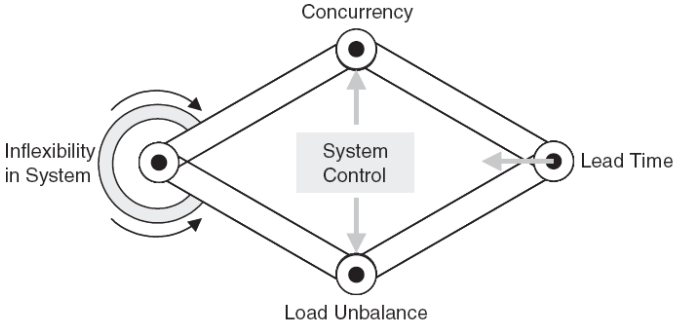

Fig. 1. Relacionamento entre flexibilidade (*Inflexibility in System*), concorrência (*Concurrency*), controle (*System Control*), desequilíbrio (*Load Unbalance*) e *lead-time* de um FMS (Wadhwa *et al*, 2005).

inovação ou até mesmo o lançamento de novos produtos. Isto significa que o sistema de manufatura também sofrerá diversas alterações. Mas, a inflexibilidade do sistema afeta o *lead-time*, tornando-o mais longo e aumentando os prazos de retorno dos investimentos. Então, observando-se novamente a Figura 1, é possível ver-se que, se a inflexibilidade for diminuída, menor será o *lead-time*, dando ao sistema maior capacidade de fabricação de diferentes produtos. Por outro lado, a complexidade do sistema de controle aumenta, pois é preciso que o sistema seja re--configurável.

Preservar a modularidade de um sistema é importante quando se quer a re-usabilidade, flexibilidade e manutenção dos projetos (Bonfe *et al*, 2002). Porém, se muitos componentes têm de se comunicar, a complexidade do modelo aumenta, afetando a flexibilidade, porque se torna mais difícil integrar os componentes. A re-usabilidade e flexibilidade de uma máquina são maximizadas se sua sincronização com o sistema global é alcançada por meio de sinais e intertravamento com um elemento supervisor. Quando a modularidade se mantém, tem-se a vantagem de prover mais flexibilidade, eficiência computacional e segurança para a aplicação de controle (Queiroz e Cury, 2002).

Um FMS possui características modulares, pois é composto, tecnologicamente, por um sistema completamente integrado, com estações de trabalho automatizadas (ex.: máquinas CNC) e/ou manuais, por recursos de manipulação e transporte, e, ainda, por sistemas de estocagem (Babic, 1999 e Gertosio *et al*, 2000). Todos esses equipamentos podem ser controlados por um ou mais computadores centrais.

Em uma visão estratégica, Abdi e Labib (2003) definem que um FMS deve ser flexível em virtude das constantes mudanças de mercado, e, também, quando for designado para tipos de produtos limitados, em condições de mercado previsíveis. Um FMS é composto de tecnologia adaptável com o objetivo de atrair mais demanda de mercado. Consequentemente, pode-se dizer que a manufatura é *puxada*, pois é a demanda que determinará o quanto deve ser produzido.

Babic (1999) discorre, ainda, sobre a complexidade de se planejar as capacidades do sistema, como operações, transporte, estoque e outras, definindo que deve haver um minucioso estudo da integração dessas capacidades. Então, o projeto do FMS, além de ser um alto investimento, incorpora uma fase cuidadosa no

sentido de garantir a captação da demanda de mercado, sendo que este está em constante atualização (Gertosio *et al*, 2000).

Do ponto de vista econômico, a implementação de um FMS é justificável quando a variedade de produtos não impõe grandes mudanças no sistema. Uma variedade muito alta significa maiores investimentos e compromisso com os riscos (Adbi e Labib, 2003), para se permitir maior flexibilidade. Então, um FMS de bom desempenho é composto por um número adequado de recursos (Babic, 1999), o que irá propiciar alta produtividade e maior flexibilidade da produção (Gertosio *et al*, 2000).

Um FMS deve ter a capacidade de produzir vários tipos de produtos sem alterações de *layout* significativas, e, também, manufaturar vários produtos ao mesmo tempo, podendo atender, paralelamente, a uma variedade de clientes. Eventualmente, faz-se necessária, diante da necessidade de se desenvolver um novo produto, a inclusão de um ou mais processos ao sistema. Isto inclui aquisição de novas máquinas e mão-de-obra. Para a inclusão desses novos recursos, o sistema deve ser re-configurável.

A metodologia proposta por Busetti e Santos (2006), que é usada como base para este trabalho, mostra um Ciclo de Desenvolvimento para sistemas re-configuráveis composto de três etapas: modelagem dos subsistemas que compõem o processo, síntese da arquitetura de controle, e implementação baseada em *Hardware-in-the-Loop* (HiL).

Na etapa de modelagem, os modelos são projetados como Sistemas a Eventos Discretos (SED), representados por autômatos. Na etapa de síntese, são obtidos supervisores modulares (autômatos que controlam o sistema) a partir das especificações do processo de manufatura em questão, visando a manter a modularidade do sistema. Pode-se dizer, também, que é uma extensão da TCS, chamada de Abordagem Modular Local (AML). A implementação é dividida em três fases, sendo que a evolução de uma fase a outra utiliza o conceito de *Hardware-in-the-loop* (Bullock *et al*, 2002).

Todas as fases da etapa de implementação seguem o conceito da proposta de Queiroz *et al* (2001), organizando o sistema em uma arquitetura de controle de três níveis, assim designados: Supervisores Modulares (SM), Sistema Produto (SP) (representação dos recursos) e Procedimentos Operacionais (PO). Este estudo está mais focado na implementação de controladores industriais

chamados Controladores Lógicos Programáveis (CLP). Porém, alguns cuidados na implementação do código em CLP devem ser tomados, ainda mais quando a TCS é usada como ferramenta de suporte a modelagem. A TCS é utilizada para a modelagem do SFC, das especificações de controle e para a síntese formal de um conjunto de supervisores. A programação em CLP é baseada nos resultados obtidos com a aplicação da TCS. Problemas de implementação, como o efeito avalanche, causalidade, entre outros, são demonstrados por Lauzon *et al* (1996). Vieira *et al* (2006) e Vieira (2007) apresentam algumas soluções, como a transformação dos autômatos em *Sequential Function Charts* (SFC).

Outra questão relacionada com a implementação da arquitetura de controle envolve a distribuição do sistema e os meios de comunicação pelos quais os equipamentos ali distribuídos irão trocar informações. Muitas são as redes usadas no meio industrial, como Profibus, DeviceNet, Modbus, Ethernet Industrial (iE – *industrial Ethernet*) e outras. A iE tem-se destacado devido à facilidade de sua compreensão, instalação e manutenção (Lee & Lee, 2002). Porém, o não-determinismo da iE é um dos principais problemas do protocolo. O atraso de comunicação na iE é um dos obstáculos a serem vencidos para implementação da TCS em CLP. Vieira (2007) propõe um modelo de comunicação confiável entre CLP aplicado à Arquitetura de Controle Supervisório (ACS), de Queiroz e Cury (2002). Tal modelo garante o envio e recebimento de eventos através de mensagens. Para a comunicação, esse modelo utiliza a troca de mensagens entre controladores, que fazem uso de blocos de comunicação definidos pela IEC 61131-5 (IEC, 2000). Adicionalmente, outros métodos de comunicação são utilizados, por exemplo, a EtherNet/IP (Allen-Bradley, 2008), que escreve ou lê a informação desejada da memória interna de outro controlador.

Foi apresentada até aqui uma introdução das diversas ferramentas formais e tecnológicas utilizadas como suporte para o projeto de sistemas de controle de sistemas flexíveis de manufatura. A seguir, serão apresentados a justificativa, os objetivos e a metodologia científica adotada para a execução deste trabalho.

1.1 Justificativa

O projeto da ACS para sistemas de manufatura precisa ser rápido e preciso, pois se sabe que é de grande importância que o tempo de projeto de sistemas de manufatura seja cada vez menor, com o objetivo de não se perder tempo em relação aos concorrentes. Porém, poucos são os trabalhos desenvolvidos para validar arquiteturas de controle implementadas em CLP através de simulação estimulada por sinais reais.

O tema se torna complexo quando se quer utilizar tecnologias industriais e ferramentas formais para projetar a ACS do sistema de manufatura, pois há uma grande variedade de tecnologias. A dificuldade está em integrá-las, uma vez que cada fabricante de tais tecnologias, muitas vezes, utiliza protocolos e sistemas proprietários.

Porém, a factibilidade do projeto é real, uma vez que a bibliografia disponível e as tecnologias de integração existentes dão suporte ao desenvolvimento de novas técnicas de integração, permitindo que a ACS possa ser implementada em CLP e testada através de procedimentos de simulação.

1.2 Objetivo Geral

O objetivo geral desta dissertação é validar a ACS implementada em CLP. Para isso, será necessário integrar a ACS a uma plataforma de simulação, seguindo o Ciclo de Desenvolvimento de Busetti e Santos (2006). A simulação, estimulada por eventos gerados a partir do CLP, permitirá que os projetos de sistemas de manufatura sejam testados computacionalmente, antes de serem implementados. Isto visa à agilidade do projeto, redução de protótipos e maior segurança antes da implementação dos componentes físicos de sistemas de manufatura.

1.3 Objetivos Específicos

Como objetivos específicos, podem ser elencados:

i) A definição de métodos para desenvolvimento de um Ambiente Dinâmico (AD) capaz de coletar eventos gerados pela ACS e enviá-los à plataforma de simulação. De forma similar, o AD também deve coletar eventos gerados pela plataforma de simulação e enviá-los à ACS.

ii) O estudo de um exemplo de sistema de manufatura. Uma ACS será projetada e testada para validação do AD.

iii) A implementação de uma plataforma de simulação com modelos virtuais referentes ao sistema de manufatura proposto, com o intuito de validar a ACS através da dinâmica virtual dos modelos, e, também, validar requisitos de produção através da simulação dos processos.

iv) Um exemplo de aplicação desenvolvido no AD para realizar a troca de eventos entre a ACS e a plataforma de simulação.

v) Conclusões sobre a importância de se desenvolverem ambientes como o AD.

1.4 Metodologia

Segundo Berto e Nakano (1998):

> "A metodologia da pesquisa provê subsídios ao planejamento e desenvolvimento sistematizado de uma investigação científica a respeito de um fenômeno observado na “realidade do mundo físico/material”. A metodologia utiliza um ou vários métodos combinados de observação, de maneira a apreender fatos e dados dessa realidade, com a intenção de entender, explicar e, se possível ou necessário, aplicá-la ou replicá-la em favor de outros eventos ou episódios semelhantes."

E, segundo Thiollent (1986):

> “O objetivo da metodologia consiste em analisar as características dos vários métodos disponíveis, em avaliar suas capacidades, potencialidades, limitações ou distorções e em criticar os pressupostos ou implicações de sua utilização. Além de ser uma disciplina, a metodologia também é considerada como modo de conduzir a pesquisa.”

Os métodos científicos envolvem experimentação e controle das variáveis, com testes-chave de repetibilidade e replicação dos resultados. Quando o problema é analisado sob o ponto de vista da engenharia, as questões do tipo ‘como’ tendem a predominar (Fleury e Nakano, 1996).

Berto e Nakano (1998) ainda colocam que a pesquisa experimental faz o teste das hipóteses através de um experimento controlado, projetado de forma a produzir os dados necessários, podendo ser realizado em laboratório. E que o modelo teórico-conceitual é o produto de reflexões a partir de um fenômeno observado ou relatado pela literatura (revisão bibliográfica). Considera-se como uma compilação de idéias e opiniões de diferentes autores, utilizando-se de simulação e modelagem matemática.

O método científico utilizado nesta dissertação é o experimental, pois a validação do AD ocorre pela implementação de um sistema de manufatura, que é testado em laboratório com tecnologias de controle e comunicação comumente utilizados no meio industrial.

Os métodos experimentais podem ser exemplificados pelo processo de validação da arquitetura de controle, pois o AD serve de meio de comunicação com a plataforma de simulação, na qual a dinâmica virtual do sistema dita se a ação de controle está correta ou não. Por outro lado, pode ocorrer de forma quantitativa no caso de averiguar aspectos relacionados aos requisitos de produção de um sistema de manufatura. Através dos relatórios gerados pela plataforma de simulação, é possível se avaliar se o sistema de manufatura está adequado, ou não, a suprir a demanda de mercado.

A pesquisa bibliográfica também está presente, pois é necessário realizar a exploração de métodos utilizados por outros autores, o que ajudou na adaptação e melhoramento dos métodos para o desenvolvimento do AD. Apesar de serem poucos os trabalhos semelhantes ao desta dissertação, há outros trabalhos que colaboram nas técnicas de integração de sistemas.

1.5 Estrutura da Dissertação

Este trabalho está dividido em cinco capítulos, conforme a Figura 2. A partir deste ponto, o trabalho está organizado da seguinte forma: o Capitulo 2 descreve a fundamentação teórica necessária para o entendimento deste trabalho. Esta fundamentação apresenta conceitos básicos sobre Sistemas a Eventos Discretos, Teoria de Controle Supervisório, Abordagem Modular Local, implementação da TCS em CLP, *Hardware-in-the-Loop* e o Ciclo de Desenvolvimento para FMS. No Capitulo 3, as propostas para desenvolvimento do Ambiente Dinâmico são descritas.

No Capitulo 4, um sistema de manufatura flexível é proposto, com o intuito de validar a existência do Ambiente Dinâmico. Esse sistema de manufatura é descrito em detalhes. Finalmente, no Capitulo 5, as conclusões, vantagens, desvantagens e perspectivas futuras sobre o tema do trabalho são apresentadas.

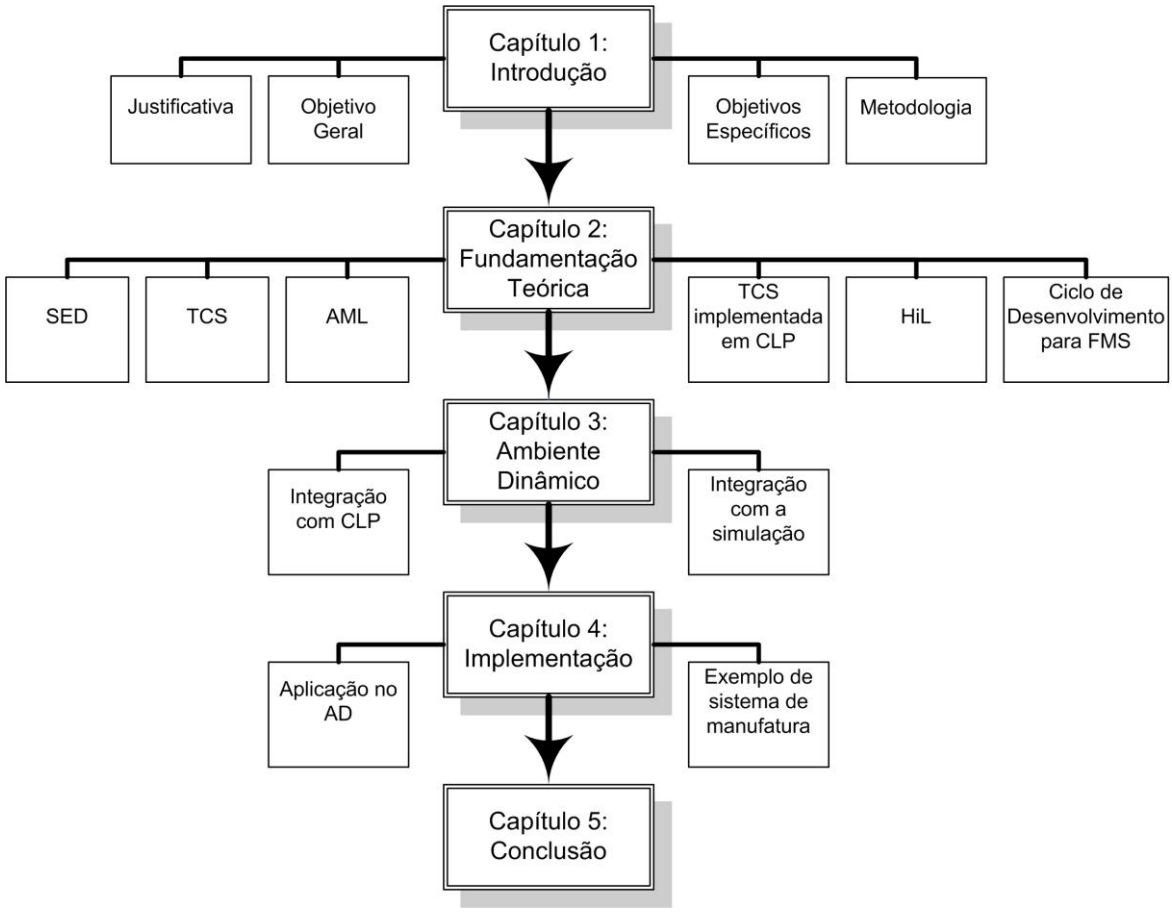

Fig. 2. Estrutura da dissertação.

1.6 Conclusão do capítulo

Este primeiro capítulo apresentou o cenário atual para desenvolvimento de arquiteturas de controle para CLP. Metodologias de implementação foram brevemente apresentadas, e as principais, para o desenvolvimento desta dissertação, serão apresentadas no próximo capítulo, como a TCS, de Ramadge e Wonham (1989) e o Ciclo de Desenvolvimento, de Busetti e Santos (2006). Os objetivos geral e específicos, bem como a justificativa, deixam clara a proposta final deste trabalho.

Mas, o mais importante a ser notado neste capítulo é que a complexidade que envolve o projeto de um FMS é o maior desafio a ser batido. Há uma ampla gama de tecnologias envolvidas, que devem ser integradas, a fim de trabalharem em conjunto para o atendimento das demandas do mercado.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta alguns conceitos fundamentais para o desenvolvimento da proposta do Capítulo 3: o Ambiente Dinâmico (AD). Nas próximas seções, estão definidos os Sistemas a Eventos Discretos (SED), a Teoria de Controle Supervisório (TCS), a Abordagem Modular Local (AML) da TCS, um método para implementação em CLP, a metodologia *Hardware-in-the-Loop* (HiL) e o Ciclo de Desenvolvimento para Sistemas Flexíveis de Manufatura (FMS). Todos esses conceitos foram utilizados para o desenvolvimento do AD, com o objetivo de validar a Arquitetura de Controle Supervisório (ACS) implementada em um Controlador Lógico Programável (CLP).

2.1 Sistemas a Eventos Discretos

Os Sistemas a Eventos Discretos (SED) percebem as ocorrências ao seu redor através da recepção de estímulos, denominados eventos. Um evento deve ser pensado como de ocorrência instantânea e como causador de uma transição no valor discreto do estado do sistema. Exemplos de eventos podem ser arrolados em:

- sistemas de filas, como a chegada de clientes em uma fila de banco;
- sistemas de computação, como o início e o fim de uma tarefa, mas não a sua execução;
- sistemas de comunicação, como a recepção de uma mensagem;
- sistemas de manufatura, como um sinal de chegada de uma peça num processo industrial ou o aperto de um botão pelo operador de uma máquina.

Adotando-se o último exemplo apresentado, pode-se dizer que um sistema de manufatura é um sistema dinâmico e sua evolução é estimulada por eventos. Por esta razão, este tipo de sistema pode exibir as características de SED (Lauzon *et al*, 1996).

> "Um Sistema a Eventos Discretos é um sistema dinâmico que evolui de acordo com a ocorrência abrupta de eventos físicos, em intervalos de tempo em geral irregulares e desconhecidos" (Cury, 2001; Ramadge e Wonham, 1989).

A ocorrência de um evento causa, em geral, uma mudança interna no sistema, que pode ser causada pela ocorrência de um evento interno do 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, onde 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 desses eventos pode depender de fatores alheios ao sistema, de modo que este não tem, em geral, como prevê-los (Cury, 2001).

Diz-se que, na 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

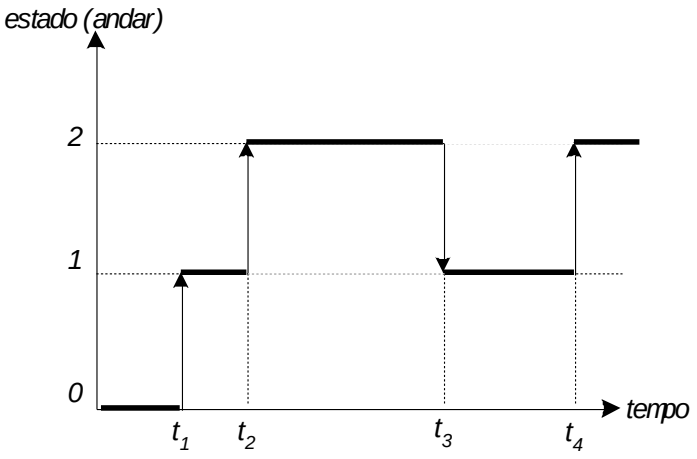

Fig. 3. Evolução típica de um SED.

descer. Supõe-se, ainda, que o elevador esteja inicialmente no térreo, e a sequência de movimentos é representada na Figura 3 (Kumar e Garg, 1995). Observa-se que o elevador tem três estados: 0, 1 e 2; na sua trajetória, ocorrem dois eventos: *subir* e *descer*. Nota-se que um mesmo evento pode ter efeitos diferentes, dependendo do estado em que ocorra, e a trajetória pode continuar indefinidamente, inclusive com a ocorrência de novos eventos ou estados. 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.

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 (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 que causam transição 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-se 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 SED, já que se caracterizam pela continuidade das variáveis de estado e cujo mecanismo de transição é dirigido pelo tempo. Ao

contrário desses sistemas, os SED podem permanecer um tempo arbitrário em um mesmo estado, sendo que sua trajetória pode ser dada por uma sequência de eventos na forma $\{\sigma 1, \sigma 2, ...\}$, incluindo, eventualmente, os instantes de tempo em que tais eventos ocorrem $\{(\sigma 1, t1), (\sigma 2, t2), ...\}$. A quantidade de informação necessária depende dos objetivos da aplicação (Cury, 2001).

A Figura 4 (Cassandras e Lafortune, 1999) ilustra as características que distinguem os Sistemas a Variáveis Contínuas dos Sistemas a Eventos Discretos (SED). Para os Sistemas a Variáveis Contínuas (Figura 4a), o espaço de estados é o conjunto dos números reais e $x(t)$ poderá assumir qualquer valor desse conjunto. Também, a função $x(t)$ é a solução de uma equação diferencial na forma geral $\dot{x}(t) = f(x(t), u(t), t)$, onde $u(t)$ é a entrada. No caso de SED (Figura 4b), 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 especifique o modo como eventos interagem com o tempo, ou como o tempo de ocorrência pode ser determinado (Cassandras e Lafortune, 1999).

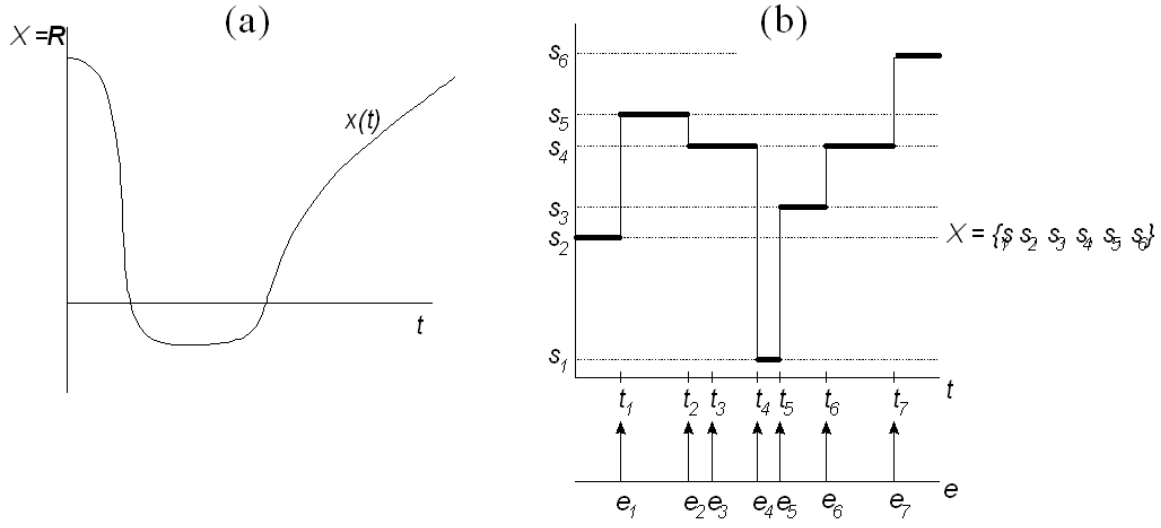

Fig. 4. (a) Sistemas a Variáveis Comtínuas e (b) SED.

2.2 Representação de SED por linguagens

Conforme Cury (2001), uma linguagem L definida sobre um alfabeto Σ é um conjunto de cadeias formadas por símbolos pertencentes a Σ. Um exemplo de linguagem sobre o alfabeto $\Sigma = \{\alpha, \beta, \gamma\}$ é $L_1 = \{\beta, \alpha, a\beta\beta\}$. O conjunto de todas as possíveis cadeias finitas compostas com elementos de Σ é denotado Σ^*, incluindo a cadeia vazia, denotada por ε. Assim, uma linguagem sobre Σ é sempre um subconjunto de Σ^*. Em particular, $\varnothing$, ε e Σ^* são linguagens. Se $tuv = s$, com $t, u, v \in \Sigma^*$, então:

- t é chamado prefixo de s;
- u é chamada uma subcadeia de s;
- v é chamado sufixo de s.

O comportamento de um SED pode ser descrito por um par de linguagens $D = (L, L_m)$, sendo o alfabeto Σ correspondente ao conjunto de eventos que afetam o sistema, $L \subseteq \Sigma^*$ o comportamento do sistema e $L_m \subseteq L$ o comportamento marcado do sistema, ou seja, as tarefas completas que o sistema pode realizar. Assim sendo, uma linguagem é uma maneira formal de descrever o comportamento de um SED.

2.3 SED modelados por autômatos

Um autômato é uma quíntupla $G = \Sigma, Q, \delta, q_0, Q_m$, sendo que os elementos dessa estrutura, respectivamente, são: alfabeto de eventos; conjunto de estados; $\delta : \Sigma \times Q \to Q$, função de transição de estados, ou seja, uma função parcial definida em cada estado de Q para um subconjunto de Σ (Queiroz e Cury, 2002;

Cury, 2001); $q_0 \in Q$, estado inicial e $Q_m \subseteq Q$, conjunto de estados marcados. Um autômato pode ser representado por um grafo orientado, no qual os nós representam os estados, por exemplo, os estados possíveis de uma máquina. E os arcos desse grafo representam a transição entre os estados. Um exemplo de grafo pode ser visto na Figura 5.

O comportamento de um SED pode ser representado por um autômato. Fazendo uma analogia com o grafo G da Figura 5, pode-se dizer que ele representa o funcionamento de uma máquina ou o de uma planta, sendo que o estado inicial, representado por um círculo de borda dupla, designa máquina em repouso, e o estado, representado por um círculo simples, máquina em operação. A transição α, que é um evento, dita o início de funcionamento da máquina, enquanto que a transição β, o fim de operação.

Um autômato está associado a duas linguagens: a gerada $L(G)$, que representa as cadeias que podem ser seguidas no autômato, partindo-se do estado inicial, e a marcada $L_m(G)$, que representa o comportamento marcado ou conjunto de tarefas completas do sistema representado pelo autômato.

2.4 Teoria de Controle Supervisório

A Teoria de Controle Supervisório (TCS) de Sistemas a Eventos Discretos (SED) foi formulada, inicialmente, por Ramadge e Wonham (1989). Essa teoria tem sido desenvolvida nas últimas décadas como uma proposta de metodologia formal de síntese de controladores ótimos para SED, entre os quais se inclui grande parte dos sistemas de manipulação e montagem automatizados.

Na abordagem proposta por Ramadge e Wonham (1989), o SED a ser controlado, ou planta, na terminologia de controle tradicional, é representado por

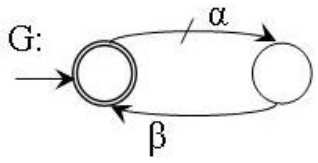

Fig. 5. Exemplo de grafo orientado.

uma linguagem gerada L (sequências parciais) e por uma linguagem marcada L_m (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 $L_m(G)$ podem conter cadeias indesejáveis de eventos, por violarem alguma condição que se deseja impor ao sistema. Pela junção de uma arquitetura de controle (supervisor), será possível modificar-se a linguagem gerada pelo sistema, dentro de certos limites, evitando-se 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 a de que esse comportamento não é satisfatório e deve ser modificado por uma ação de controle. Tal modificação 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 desabilite num dado instante dependam do comportamento passado do SED. Esta abordagem é referida como controle supervisório monolítico, pois o objetivo é projetar um único controlador, cuja função seja habilitar e desabilitar certos eventos, conforme a sequência 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, terão ocorrência permitida a seguir. Sob esse aspecto, a forma de controle é dita permissiva, no sentido de que eventos inibidos não podem ocorrer, e, os autorizados, não precisam obrigatoriamente ocorrer. O conjunto de eventos habilitados num dado instante pelo supervisor define uma entrada de controle, que é atualizada a cada nova ocorrência de evento observada em G. A Figura 6 ilustra o acoplamento entre a planta e o supervisor.

Esses 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.

Para se associarem arquiteturas de controle a um SED ou a uma planta G, particiona-se o alfabeto Σ em um conjunto Σ_c de eventos controláveis que podem ser inibidos de ocorrer, e um conjunto Σ_u de eventos não-controláveis, sobre os quais o agente de controle não tem influência. Para que seja possível se interferir no funcionamento da planta G, este funcionamento precisa ser dotado de uma interface através da qual se possa informar quais eventos devem ser habilitados e quais devem ser desabilitados. 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 arquitetura de controle associada a G como o conjunto de entradas de controle:

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

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 de Ramadge e Wonham (1989), 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-

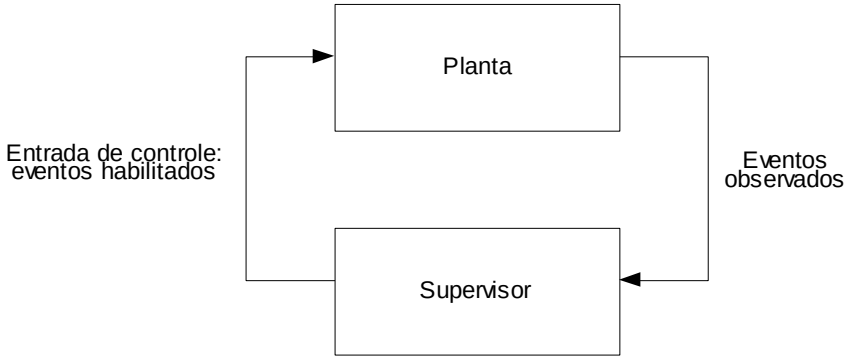

Fig. 6. Acoplamento entre planta e supervisor.

controláveis, deixa de ser definida para os eventos desabilitados 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, satisfaz:

$$i) \varepsilon \in L(f/G) \text{ e}$$

$$ii) w\sigma \in L(f/G) \text{ sse } w \in L(f/G), w\sigma \in L(G) \text{ e } \sigma \in f(w).$$

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 entradas 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:

$$L(S/G) = L(S \| G) \text{ e}$$

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

De um modo geral, um problema de síntese de supervisores supõe que se represente, por linguagens, o comportamento fisicamente possível do sistema e o comportamento desejado sob supervisão, 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 arquitetura 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 os 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. Podem ser certos estados de G que sejam indesejados e devam ser evitados, por serem causadores de bloqueio ou então fisicamente inadmissíveis, como, por exemplo, a colisão de um robô com um veículo auto-guiado, ou a tentativa de se colocar uma peça num *buffer* cheio num sistema de manufatura automatizado. Ou, ainda, algumas palavras em $L(G)$ podem conter prefixos que não sejam permitidos e violam uma sequência desejada de certos eventos. Assim, consideram-se sublinguagens de $L(G)$ que representam o comportamento “legal” ou “admissível” do sistema controlado (Cassandras e Lafortune, 1999).

2.5 Abordagem Modular Local da TCS

A Teoria de Controle Supervisório (TCS), proposta por Ramadge e Wonham (1989), apresenta a vantagem de permitir a síntese automática de

supervisores, e, também, a noção de máxima linguagem controlável, que garante a síntese de controladores de forma minimamente restritiva. No entanto, quando um grande número de tarefas deve ser executado pela arquitetura 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 se diminuir a complexidade da síntese de controladores consiste em se dividir a tarefa de controle em várias subtarefas, que são resolvidas usando-se a teoria de controle segundo Ramadge e Wonham (1989), e em se combinar os subcontroladores resultantes de modo a solucionar o problema original. Esta concepção é denominada síntese modular, e, os controladores resultantes, supervisores modulares (SM). Esta abordagem foi introduzida por Wonham e Ramadge (1988), sendo 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 subtarefa for mudada, basta apenas se reconstruir o sub-controlador correspondente, ao invés de se 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 arquitetura de controle e da complexidade do processo de síntese. O conceito de modularidade é apresentado a seguir.

Sejam duas linguagens $L_1, L_2 \subseteq \Sigma^*$. 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 L_1 e L_2 é também um prefixo de L_1 e L_2. Diz-se que L_1 e L_2 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 compartilharem um prefixo, compartilharem 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 arquitetura 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.

Queiroz e Cury (2000a, 2000b) 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 Abordagem Modular Local (AML), com uma arquitetura distribuída em que cada módulo de controle atua somente sobre os subsistemas atingidos. Dessa forma, o controle modular proposto por Queiroz e Cury (2000a, 2000b) 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 arquitetura de controle. É esta a abordagem adotada nesta dissertação.

2.6 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, menor espaço de memória e costuma ser mais compreensível para o projetista.

Tais sistemas são normalmente modelados pela composição de subsistemas de menor porte, podendo tais subsistemas ser assíncronos entre si.

Dessa forma, de acordo com o nível de composição das subplantas originalmente modeladas, diferentes representações para o sistema podem ser formuladas. Na AML, são utilizadas duas representações para o sistema: a Representação por Sistemas Compostos (RSC) e a Representação por Sistemas-Produto (RSP) (Ramadge e Wonham, 1989). A definição dessas duas representações é dada a seguir.

2.6.1 Representação por Sistema Composto

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

2.6.2 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 Wonham, 1989; Ramadge, 1989). Denomina-se Representação por Sistema-Produto (RSP) qualquer RSC cujas subplantas não tenham eventos síncronos em comum.

Considerando-se que cada subsistema de uma RSP representa uma parte isolada de um sistema em malha aberta, pode-se afirmar que essa modelagem representa a estrutura descentralizada natural de operações concorrentes para sistema de maior porte. Pode-se obter a mais refinada RSP, equivalente a uma RSC qualquer, pela composição dos geradores síncronos desta última. 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 $G_{Xj} = \|_{i \in N_{Xj}} G_i$, com $N_{Xj} = \{k \in N \mid \Sigma_K \cap \Sigma_{xj} \neq \varnothing\}$. Assim, a planta local G_{Xj} é composta apenas pelos subsistemas da modelagem original que estão diretamente (e indiretamente) restringidos por E_{xj}.

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 7. 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 $E_B = E_b \| L_m(G_B)$.

2.7 Arquitetura de Controle Supervisório

Considera-se que a Arquitetura de Controle Supervisório (ACS) é implementada em CLP. 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

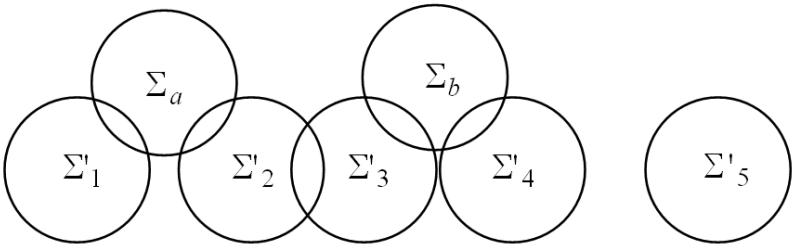

Fig. 7. Alfabeto de eventos de um sistema composto.

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 ACS é desenvolvida em três níveis estruturais: o nível dos Supervisores Modulares (SM), que desabilita eventos de acordo com as mudanças de estado da planta; o nível de Sistema-Produto (SP) que, seguindo os modelos supervisionados das plantas, é responsável por comandar o início das sequências de operação; e o nível dos Procedimentos Operacionais (PO), que, observando os sinais de entrada do CLP e ajustando os sinais de saída, executa os ciclos de funcionamento de cada dispositivo. A ACS é ilustrada na Figura 8. O modelo apresentado por Queiroz *et al* (2001) também é empregado no presente trabalho, com algumas extensões apresentadas no Capítulo 3.

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 SP sem que os supervisores tenham sido atualizados.

Apesar de a ACS prover uma distribuição da implementação dos SM e do SP, um único CLP é usado neste caso. Como os sistemas de manufatura são fisicamente distribuídos na planta, um novo método foi proposto por Vieira *et al*

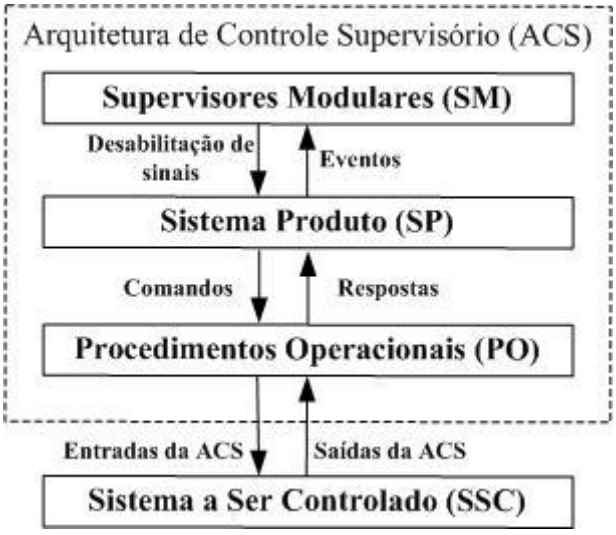

Fig. 8. Arquitetura de Controle Supervisório proposta por Queiroz *et al* (2001).

(2006) e Vieira (2007). Este método prevê uma ACS que permite a inserção de CLP dispostos em uma rede, conforme Figura 9. Há, então, um CLP coordenador com os SM e o SP, e, no mínimo, outro CLP executor com PO.

Este modelo de ACS é usado nos três passos da etapa de implementação do Ciclo de Desenvolvimento proposto por Busetti e Santos (2006). Este mesmo modelo de ACS está presente no Ambiente Dinâmico (AD), pois os sinais referentes a comandos e respostas (entre os níveis SP e PO) são lidos e escritos pelo AD, fornecendo informações para a evolução da simulação.

Baseado na ACS proposta por Queiroz *et al* (2001), Vieira (2007) propõe um método de implementação da ACS em CLP. Este método permite que o projetista converta, sistematicamente, o conjunto de supervisores e o conjunto de autômatos na síntese dos supervisores, em um conjunto de *Sequential Function Charts* (SFC). O programa de CLP obtido está de acordo com a IEC 61131-3 (2003).

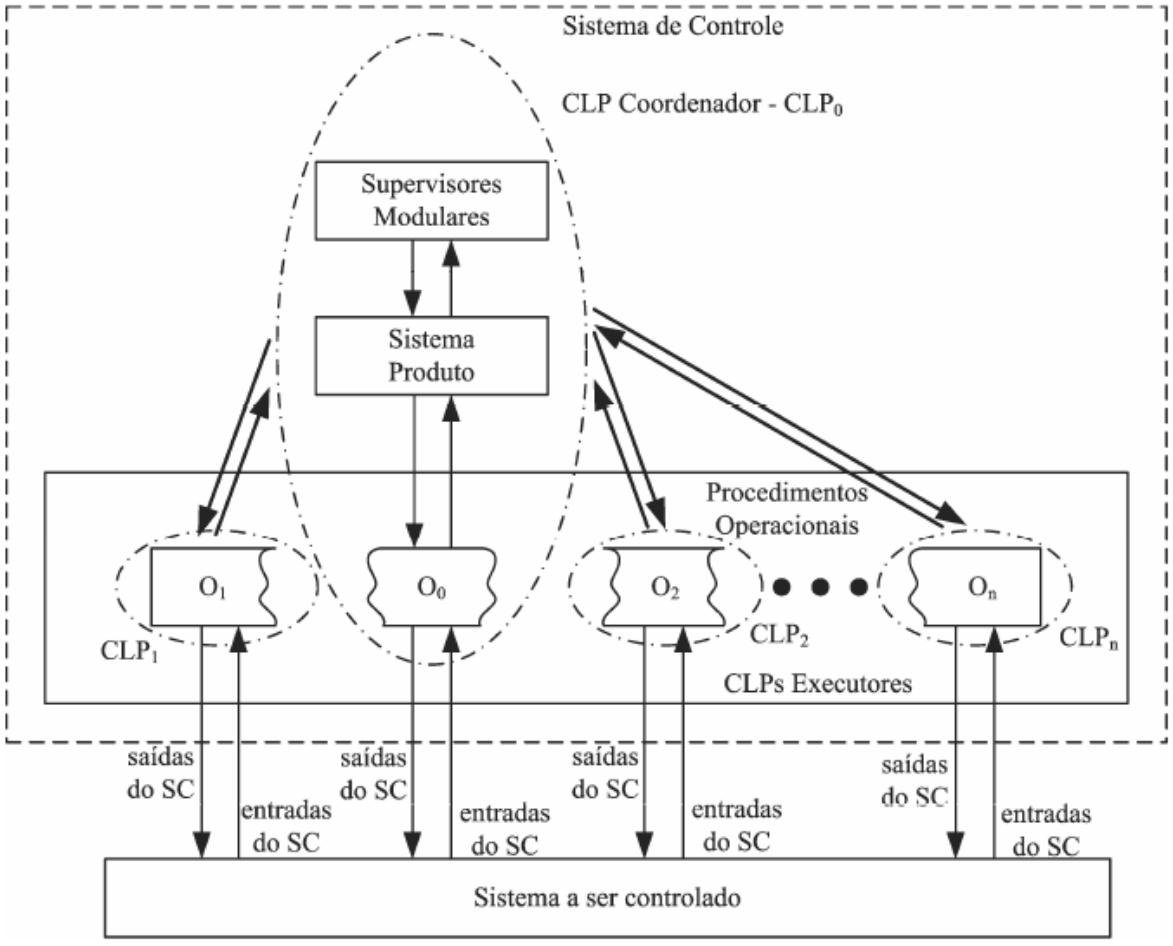

Fig. 9. ACS distribuída em *n* CLP (Vieira, 2007).

2.8 Modelo para implementação distribuída do controle supervisório

Primeiramente, Vieira (2007) propõe um modelo de implementação concentrada, e, posteriormente, uma distribuição da implementação em um conjunto de CLP. Neste último caso, o autor visa a incrementar o desempenho do sistema, além de realizar a distribuição espacial do controle, integrar controladores, viabilizar a utilização desses controladores considerando a limitação dos mesmos, possibilitar a reutilização das estratégias de controle, simplificar a reconfiguração e preservar a modularidade do sistema a ser controlado. A distribuição da ACS em diversos controladores pôde ser visualizada na Figura 9. Para a programação dos CLP, Vieira (2007) organiza seu modelo em diversos *Function Blocks* (FB), que são explicados no decorrer desta seção.

Vieira (2007) propõe um SFC, visualizado na Figura 10, denominado de *SFC Main*, que estabelece o modo de operação do sistema de controle. Considera-se que o passo *init* é ativado na primeira execução do programa. Quando o usuário ativa a variável *Sinit,* o passo *SI* é ativado, significando que o programa está no modo de inicialização do sistema de controle. *PSinit* é uma entrada física que permite a inicialização do sistema, a ser controlado através do passo *PSI*. Quando a inicialização está pronta, a variável *PSReady* fica ativa, fazendo com que o SFC evolua para o passo *PSIted*, no qual o sistema espera uma decisão do usuário, se funcionará em modo manual (*Man*) ou supervisionado (*Sup*). O passo *Sup* realiza, através da ação *action Sup,* a coordenação da operação dos diversos subsistemas.

Os autômatos que representam o comportamento dos subsistemas são traduzidos para SFC em três etapas. Cada SFC referente a um subsistema ocupa a seção de código de um FB; portanto, haverá um FB para cada subsistema. A primeira etapa define variáveis que serão utilizadas no FB onde estará implementado o SFC. A segunda, permite converter o autômato de um subsistema em outro autômato que satisfaz um conjunto de propriedades de interesse. A terceira etapa define o SFC do subsistema.

Toda vez que um evento é tratado no FB de um subsistema, e este subsistema compartilha uma célula de controle com outro subsistema, o tratamento de eventos deve ser suspenso temporariamente, até que todo o estado ativo do supervisor seja atualizado com o evento tratado. Isto é necessário para atualizar a ação de controle dos supervisores antes do tratamento de outros eventos. Este procedimento de suspender o tratamento de eventos é realizado por um FB designado a cada FB do subsistema correspondente.

Outros FB propostos são os que contêm a seção de código referente a cada SM. Estes são obtidos na forma de autômatos, sendo necessária a tradução dos mesmos para SFC sistematicamente, através de duas etapas. A primeira, define as variáveis utilizadas no FB, e, a segunda, define os SFC correspondentes.

Vieira (2007) define um FB denominado *MS* que faz a chamada sequencial de todos os FB onde estão implementadas as seções de código referentes aos SM. Este procedimento permite a atualização dos SM e a definição

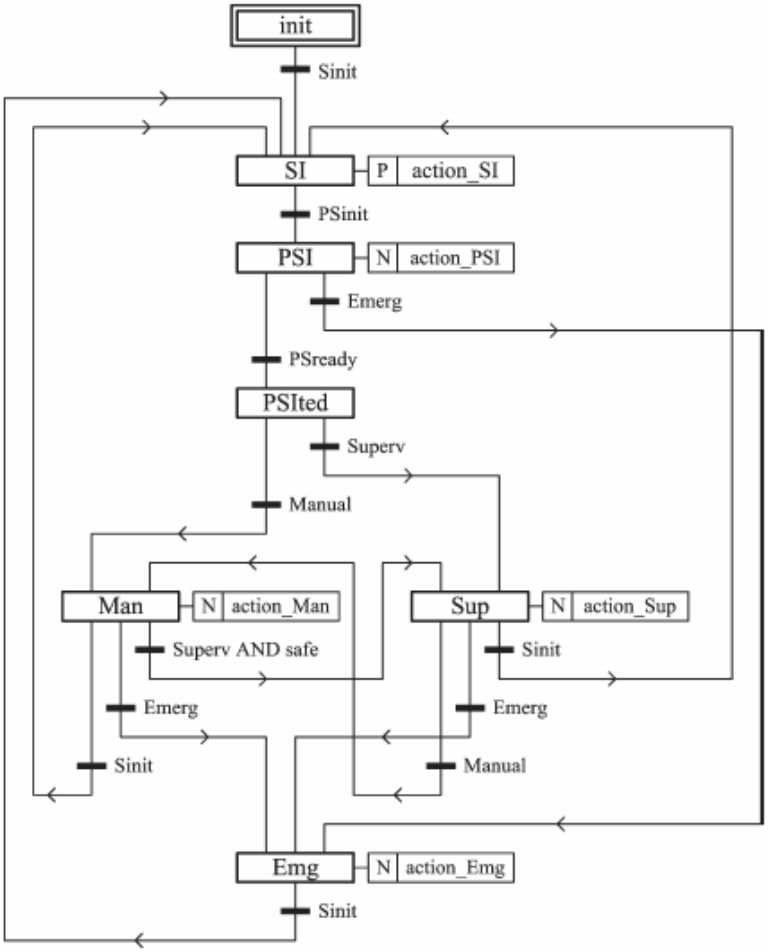

Fig. 10. ACS distribuída em *n* CLP (Vieira, 2007).

dos sinais de desabilitação. Outro FB definido é o *PS*, que realiza a chamada sequencial de todos os FB de suspensão do tratamento de eventos imediatamente antes de chamar os FB onde estão implementadas as seções de código referentes aos subsistemas. Alguns FB não são explanados aqui, pois não têm importância para o desenvolvimento do presente trabalho.

A chamada dos FB do parágrafo anterior são feitas pela ação *action_Sup* do SFC *Main*. Esta ação executa uma sequência de atividades, como a chamada do FB *MS* para realizar a atualização do estado ativo dos supervisores, e a ação de controle. Outra ação é a desativação da variável de sinalização de evolução de um subsistema, pela negação de um comando relacionado ao evento controlável do subsistema. Mais uma ação é a suspensão de todo evento controlável para priorizar eventos não-controláveis, sendo revertida uma vez que os eventos não-controláveis tenham sido tratados pela chamada do FB *PS*.

O funcionamento destes FB pode ser entendido mais facilmente no Capítulo 4, em que é implementado um exemplo hipotético de sistema de manufatura.

2.9 Hardware-in-the-Loop

Várias definições de HiL são encontradas na literatura. Plummer (2005) considera HiL como uma simulação executada em tempo real , com a função de exercer controle sobre uma planta. Já Misselhorn *et al* (2006) acreditam que o conceito de HiL é usado como um método de teste, para posteriormente ser usado como integração de peças mecânicas através de *softwares* de simulação. Stoeppler *et al* (2005) afirmam que a simulação HiL é tal, que o sistema simulado está ligado ao sistema de controle real, de maneira que toda simulação possa operar de forma similar ao comportamento real. De uma forma geral, HiL é uma técnica que combina e interconecta real e virtual, dentro de uma configuração operacional, para simular e testar o comportamento dinâmico de sistemas complexos (Cravotta, 2005). A simulação HiL é utilizada neste trabalho para validação dos modelos dos

subsistemas de manufatura e das especificações comportamentais, porém não exercerá poder de controle sobre a planta.

Loures (1999) utiliza HiL na realização experimental de seu trabalho. Ele afirma que HiL tem sido amplamente utilizado como ferramenta de suporte ao desenvolvimento de sistemas de controle. E também é encontrado em sistemas mecatrônicos, devido à complexidade da planta que os compõem. A planta é dividida em subsistemas, que são inicialmente representados computacionalmente (modelo matemático). No decorrer do projeto, os subsistemas computacionais são substituídos por subsistemas reais (modelo real), conforme ilustrado na Figura 11. O ambiente de simulação, no qual são feitas as análises para verificação de possíveis erros e ajustes, é o mesmo ambiente da simulação em tempo real.

A configuração de um sistema de controle, com adoção de simulação HiL, permite a interface entre os modelos projetados e os elementos físicos, proporcionando validação de tais modelos (Loures, 1999). A definição das técnicas depende dos objetivos propostos e das características do sistema a ser implementado. No entanto, o processo *off-line* e *on-line* sempre estarão presentes num projeto que utiliza HiL. O último processo é a simulação em tempo real. Como Loures (1999) utiliza as técnicas de HiL para validação de modelos de controle de processos, as exigências desses sistemas pedem que a simulação seja realizada em tempo real, com requisitos rígidos.

É possível simular e testar o sistema de manufatura antes da sua implementação em tempo real, substituindo, retirando e realocando subsistemas

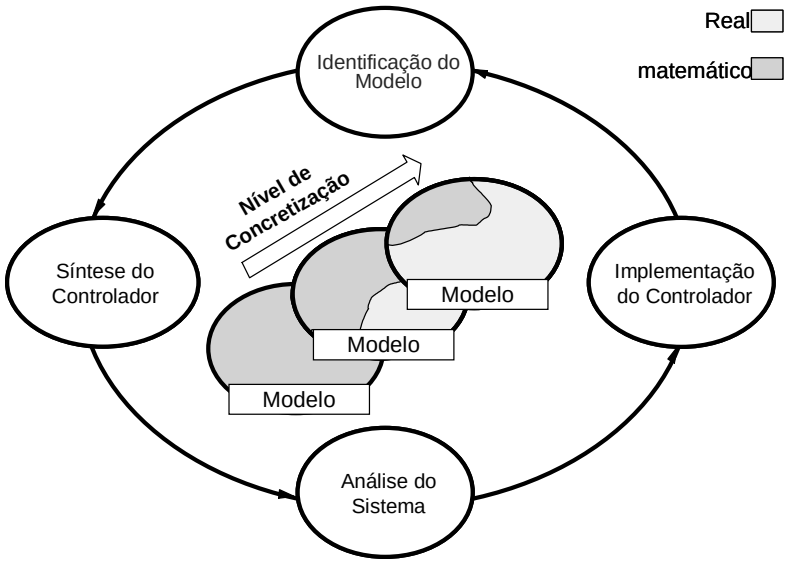

Fig. 11. Ciclo de Desenvolvimento para simulação com HiL (Loures, 1999).

menores. Uma das maiores vantagens do HiL é que o sistema, ou partes dele, podem ser testados virtualmente até os limites mais extremos, sem que haja perigo de se danificar essas partes. A simulação de pequenas porções do sistema é viável, mesmo que o sistema não esteja completamente pronto (Kocijan e Karba, 1995). Ou ainda, como o sistema simulado em tempo real responde a sinais do sistema físico, a simulação de um subsistema é viável antes que seja incorporado ao sistema real (Cravotta, 2005).

Outro fator positivo é que a análise do que ocorre no sistema torna-se uma importante ferramenta no que diz respeito à tomada de decisão, pois, através de repetições da simulação, obtêm-se gráficos e tabelas que podem fornecer dados concretos sobre o sistema a ser implementado.

De uma maneira geral, o HiL fornece grande suporte à integração gradual do projeto do sistema. Também, minimiza o tempo de projeto, desde sua concepção até o comissionamento da planta real (Kocijan e Karba, 1995).

Quanto maior o nível de detalhamento da simulação, maior será o custo computacional (Kocijan e Karba, 1995). Os sistemas operacionais mais comuns (Windows e Unix), sendo executados nos *hardwares* existentes, podem não ter capacidade suficiente para implementação em tempo real (Loures, 1999). Contudo, a cada dia, o *hardware* que suporta os sistemas operacionais está ficando mais potente, em termos de desempenho. Então, espera-se que, em pouco tempo, a simulação em tempo real torne-se mais viável em termos de investimento em *hardware*. Por enquanto, a melhor opção, em termos de desempenho para obter-se tempo real, é o investimento pesado em *hardware*.

2.9.1 Estrutura Básica para HiL

Segundo Bullock *et al* (2002), três elementos são essências para uma estrutura que utiliza HiL (Figura 12):

1. *Controlador de Interface (CID – Controller Interface Device):* este dispositivo permite a ligação física entre o sistema de

controle e o computador no qual será realizada a simulação, tanto *off-line* quanto em tempo real. O CID possui sinais discretos de entrada e saída.

2. *Interface de Software:* promove o encapsulamento dos sinais físicos para serem usados na simulação. É a ligação entre o CID e a plataforma de simulação.
3. *Plataforma de Simulação:* os sinais coletados através do CID são usados apenas para a simulação do sistema. A implementação da simulação do sistema em tempo real não deve exercer nenhuma ação de controle sobre todo o sistema.

Adotando-se um sistema de manufatura como exemplo, pode-se dizer que o CID é o CLP, a Interface de Software, um *software* SCADA, e, a Plataforma de Simulação, um *software* de simulação.

Há três importantes aspectos para se entender como o HiL trabalha (Bullock *et al*, 2002). Para isto, é necessário o entendimento da programação e sequenciamento das tarefas:

1. para assegurar a sincronização entre o controlador externo e a simulação, esta deve ser executada em tempo real;
2. a simulação precisa ser executada em passos uniformemente

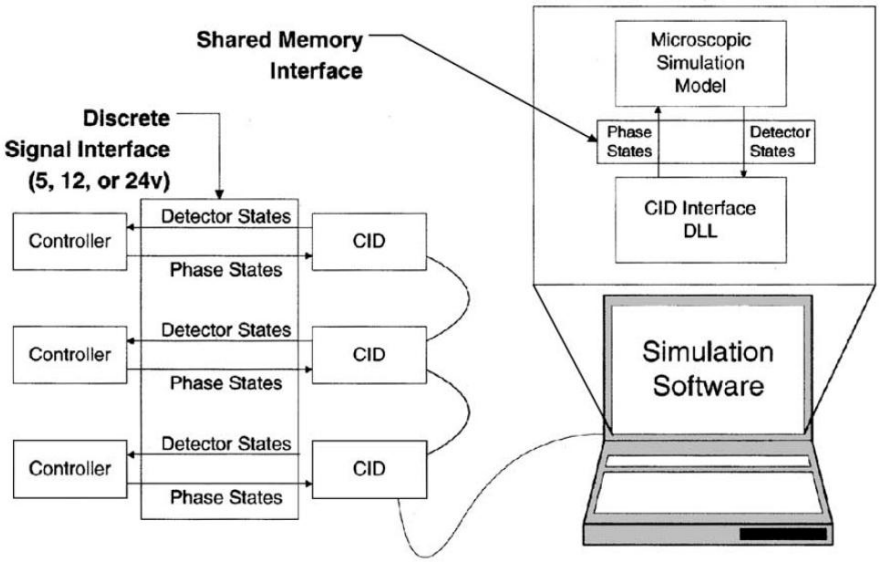

Fig. 12. Estrutura básica para HiL (Bullock *et al*, 2002).

espaçados no tempo;

3. a simulação de cada passo deve ser executada mais rapidamente do que o tempo real, para que a interface de *software* tenha tempo de ser executada e aguardar que o relógio de tempo real alcance o inicio do próximo período de simulação.

Quando a Plataforma de Simulação está operando com o *hardware* desconectado, o modelo de simulação é executado em intervalos de tempo tão curtos quanto possível, sem se preocupar com o relógio de tempo real. No entanto, quando o hardware é conectado, a sincronização entre o programa de simulação e o sistema de controle real é uma grande preocupação. Se a simulação é mais lenta que o sistema real, a simulação HiL torna-se inviável, porque é virtualmente impossível modificar o controlador real para operar mais lentamente. Já, se a simulação ocorre mais rapidamente que o sistema real, a simulação HiL torna-se viável.

A validação do estudo de Bullock *et al* (2002) ocorreu usando como aplicação um complexo sistema de semáforos de tráfego urbano. Sua simulação HiL exigiu, então, que o sistema simulado fosse executado em tempo real. Entretanto, há ocasiões em que a simulação, ocorrendo em tempo menor do que o tempo do sistema real, já é suficiente, como em casos onde apenas a informação do que ocorre no processo é o bastante.

2.10 Ciclo de Desenvolvimento para um FMS

A metodologia proposta por Busetti e Santos (2006) consiste de um Ciclo de Desenvolvimento para processos reconfiguráveis, composto por três etapas, conforme ilustrado na Figura 13. Tais etapas são:

1. Modelagem.
2. Síntese.

3. Implementação.

Este Ciclo de Desenvolvimento é usado, nesta dissertação, como base de desenvolvimento da Arquitetura de Controle Supervisório (ACS), para um exemplo de sistema de manufatura. Esta metodologia foi desenvolvida com o intuito de projetar uma ACS para sistemas reconfiguráveis, almejando a flexibilidade e agilidade desses sistemas. Isto consiste em atender os projetos de manufatura automatizados com alta eficiência, eficácia e confiabilidade, quando novas aplicações são necessárias. Há a exigência para a reconfiguração do processo, desde a inserção de novos produtos, devido às demandas de mercado e à modernização tecnológica do processo. A inserção de novos modelos é prevista no Ambiente Dinâmico (AD).

Observa-se, na Figura 13, a Biblioteca TMA. Esta biblioteca de Tecnologias de Manufatura Avançadas (TMA) fornece suporte à base tecnológica para a definição dos Procedimentos Operacionais (PO) relacionados com os subsistemas, para a implementação de Supervisores Modulares (SM) e para as necessidades de reconfiguração do sistema. Tecnologias de comunicação, CLP, sensores, atuadores e outros são exemplos de TMA e são utilizados nos sistemas

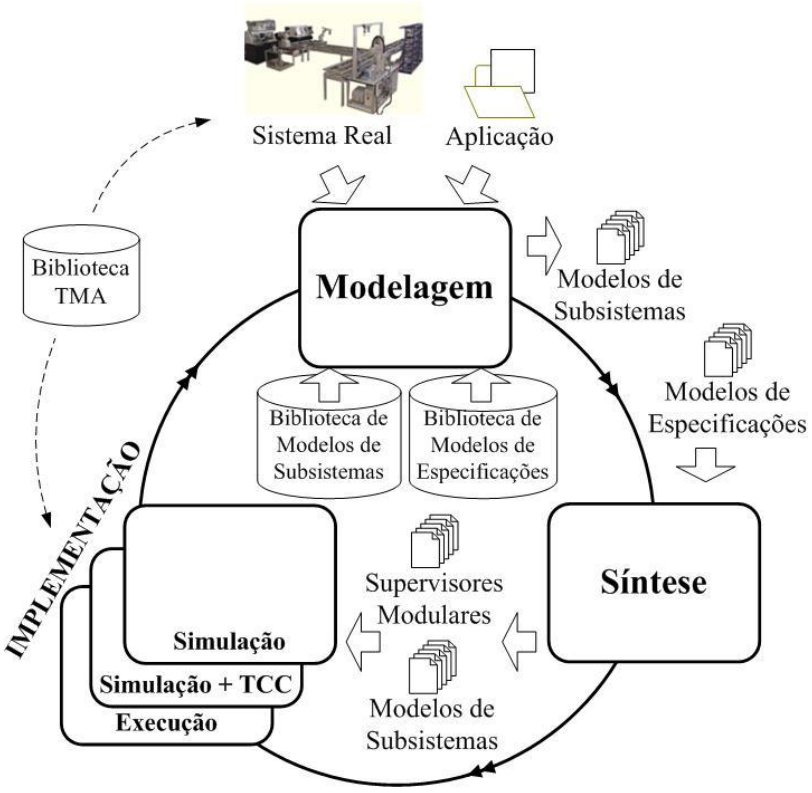

Fig. 13. Ciclo de Desenvolvimento (Busetti e Santos, 2006).

de manufatura atuais. Para o exemplo a ser estudado neste projeto, CLP e tecnologias de comunicação serão aplicados.

Nas seções a seguir, as etapas do Ciclo de Desenvolvimento são descritas, exemplificando a aplicabilidade ao AD.

2.10.1 Modelagem

Nesta etapa, os vários subsistemas podem ser modelados através de autômatos assíncronos. Desta maneira, torna-se possível obter a Representação por Sistema-Produto (RSP). Então, Busetti e Santos (2006) colocam quatro passos que devem ser seguidos:

1. Identificar os subsistemas envolvidos no sistema de manufatura.
2. Construir os autômatos G_i de cada subsistema envolvido, tão sistematicamente quanto possível.
3. Calcular a mais refinada RSP e produzir a composição de subsistemas síncronos.
4. Modelar as especificações isoladamente, considerando somente os eventos relevantes.

Na etapa de Modelagem, os subsistemas e as especificações de comportamento são modelados individualmente, permitindo que o sistema seja flexível. Quando um subsistema deixa de ser necessário ao processo, é retirado do sistema, porém armazenado em uma biblioteca de modelos ou de especificações, pois, futuramente, tal subsistema pode ser novamente inserido no sistema. Por outro lado, novos subsistemas podem ser adicionados ao processo produtivo, gerando, assim, novos modelos. Ambas as bibliotecas de modelos de subsistemas e especificações podem ser vistas no Ciclo de Desenvolvimento da Figura 13.

Uma biblioteca de modelos também é prevista no AD, para leitura e escrita de eventos entre a ACS e a Plataforma de Simulação. A partir da modelagem

dos subsistemas, é possível se dizer quais serão os eventos a serem gerados e lidos pela simulação, que deverá responder, com a escrita de outros eventos, à ACS.

2.10.2 Síntese

Um controlador monolítico é projetado de tal maneira que desabilite eventos cuja ocorrência é indesejável em uma planta, com o intuito de organizar a sequência de eventos que devam ocorrer conforme as regras operacionais especificadas. Representado por um gerador S, o supervisor tem seus estados alterados pela ocorrência de eventos na planta G. O supervisor desabilita os eventos de G que não possam ocorrem em S, após observar uma cadeia de eventos. A composição síncrona $S||G$ descreve o comportamento obtido sob ação de supervisão. Além de restringir o comportamento da planta, o supervisor tem a função de especificar as tarefas consideradas como completadas, fazendo com que uma tarefa do sistema seja completada somente se for marcada pelo supervisor e pela planta.

Este processo de síntese pode não ser aplicável em determinados processos de manufatura devido à grande quantidade de especificações e de geradores que representam as máquinas. Isto porque o número de estados cresce exponencialmente com o número de elementos do sistema, inviabilizando a implementação.

Para proporcionar um menor esforço computacional de controladores para sistemas de manufatura, a partir de cada especificação pode-se construir um supervisor modular não-bloqueante e que desabilite os eventos controláveis não desejados. A ação de controle é atribuída a cada estado de um SM como um conjunto de eventos que devem ser obrigatoriamente desabilitados.

Os seis passos sugeridos por Busetti e Santos (2006) para esta etapa, basicamente, são passos que visam aos procedimentos de síntese propostos por Ramadge e Wonham (1989) e Queiroz e Cury (2002):

1. Obter a planta local de cada especificação. Para isto, é necessário compor os subsistemas da RSP que têm eventos em comum em relação à especificação.
2. Para cada planta local, calcular a linguagem que garanta a especificação.
3. Calcular a máxima linguagem controlável contida em cada especificação local.
4. Checar a modularidade local das linguagens resultantes.
5. Se não for verificada a modularidade, tentar resolver o problema usando outra aproximação.
6. Caso seja verificada a modularidade, então implementar um supervisor local para cada linguagem controlável.

Os modelos de SM não são aplicados ao AD, pois apenas proíbem ou permitem a geração de eventos ao SP. Mas, é de suma importância que sejam implementados em CLP. Assim, será possível se avaliar, através da simulação, se as especificações projetadas estão corretas. Caso haja erros, será possível corrigi-los antes de se implementarem os subsistemas físicos.

2.10.3 Implementação

A implementação possui as características de HiL. Uma vez que se têm os sistemas modelados e os supervisores sintetizados, primeiramente, os modelos vão sendo simulados. Em conjunto com a implementação do sistema físico, a simulação em tempo real deve crescer na mesma proporção, até que todo o sistema e a simulação em tempo real também estejam implementados.

Na primeira fase da Implementação, pela proposta de Busetti e Santos (2006), os três níveis da ACS (SM, SP, PO) são simulados a fim de se verificar se os modelos virtualmente implementados estão de acordo com o funcionamento esperado. A Plataforma de Simulação contém os subsistemas de acordo com a configuração real do sistema de manufatura, e a simulação deve

ocorrer de acordo com as restrições impostas pela ACS. O resultado da simulação permitirá a detecção de erros e necessidades de modificações, para posterior validação dos modelos construídos. Nesta fase, o AD deverá ler e escrever eventos na ACS e na Plataforma de Simulação.

Na segunda fase, os subsistemas inicialmente simulados vão sendo gradualmente substituídos pelos componentes reais do sistema de manufatura. Então, faz-se necessária a inserção de uma tecnologia de comunicação para integração da Plataforma de Simulação à planta junto ao sistema de controle (TCC – Tecnologias de Comunicação e Controle). No meio industrial, são usadas as redes industriais de comunicação para tal integração. Esta integração é necessária para a validação da ACS em termos de *software* e para verificar a distribuição física do sistema de controle em termos de *hardware*. Nesta fase, o AD deverá trocar (ler e escrever) eventos entre a ACS e a plataforma de simulação, e, também, entre a Plataforma de Simulação e os subsistemas físicos já implementados, no caso de simulação paralela ao ambiente real. Para sistemas físicos já implementados, os eventos são enviados ou recebidos diretamente pelos PO, sem passar pelo AD.

Na terceira fase, todos os subsistemas devem estar implementados fisicamente nos dispositivos do sistema de manufatura. Nesta fase, os modelos construídos têm de estar traduzidos para as linguagens que cabe a cada dispositivo capaz de receber e controlar, respectivamente, os sensores e atuadores da planta. Uma destas linguagens pode ser a de CLP (IEC 61131-3, 2003). Nesta fase, a AD troca (lê e escreve) eventos apenas entre a Plataforma de Simulação e os subsistemas físicos, no caso de simulação paralela ao ambiente real.

2.11 Conclusão do capítulo

Este capítulo apresentou as diversas metodologias adotadas neste trabalho para desenvolvimento da proposta do capítulo 3. A Teoria de Controle Supervisório usada para Sistemas a Eventos Discretos é uma ótima ferramenta para modelagem de subsistemas de manufatura e seus respectivos supervisores. Outras técnicas, como Abordagem Modular Local, ajudam a diminuir o esforço

computacional dos controladores industriais. As metodologias de sistematização, como o HiL e o Ciclo de Desenvolvimento, ajudam a criar passos para se desenvolverem as arquiteturas de controle mais facilmente.

Pode-se ver, claramente, a complexidade de integração de todas as metodologias e tecnologias apresentadas neste capítulo. Porém, o resultado positivo de tal integração pode propiciar novos métodos visando à melhor eficiência em se projetarem sistemas flexíveis de manufatura.

3 DESCRIÇÃO DO AMBIENTE DINÂMICO

Este capítulo descreve o Ambiente Dinâmico (AD) usado no suporte à etapa de implementação do Ciclo de Desenvolvimento proposto por Busetti e Santos (2006).

Primeiramente, os autômatos correspondentes aos supervisores e aos modelos de subsistemas obtidos são implementados na Arquitetura de Controle Supervisório (ACS), de acordo com Queiroz e Cury (2002). Esta arquitetura é programada em um Controlador Lógico Programável (CLP) Central, de acordo com o método de implementação proposto por Vieira *et al*. (2007).

Pelo Ciclo de Desenvolvimento, no primeiro passo da etapa de Implementação, os três níveis da ACS são simulados. Porém, parte da ACS (Supervisores Modulares e Sistema-Produto) já está implementada em um CLP Central. O objetivo, então, é conectar o CLP Central a uma Plataforma de Simulação, onde estão implementados os Procedimentos Operacionais Virtuais (POV) e o Sistema Virtual a Controlar (SVC), para que ambos troquem sinais (comandos e respostas). Nesta plataforma, os subsistemas são implementados de acordo com a configuração do sistema real, utilizado para a geração dos modelos do Sistema-Produto (SP). Assim, a simulação ocorre de acordo com as restrições impostas pelos Supervisores Modulares (SM). Os resultados da simulação permitem se avaliar se os modelos de subsistemas e especificações possuem erros. A simulação da ACS é útil para realizar a primeira validação dos modelos construídos (subsistemas e especificações) ou detectar a necessidade de modificações e inclusões. O projetista pode implementar os níveis SM e SP, e, então, acompanhar a evolução de estados e ações de controle associadas.

No segundo passo da etapa de implementação, os subsistemas virtuais implementados na Plataforma de Simulação são progressivamente substituídos por componentes reais da planta, que executam os Procedimentos Operacionais Reais (POR) (ex.: CLP, etc). Desta forma, o CLP Central se comunica simultaneamente com a Plataforma de Simulação e com os subsistemas reais pela inserção de Tecnologias de Controle e Comunicação (TCC). Neste passo, é possível validar-se progressivamente a ACS.

No terceiro passo, os dispositivos de campo (ex.: CLP) onde estão implementados os POR, estão completamente conectados aos respectivos sensores e atuadores do sistema real. Neste ponto, o sistema de manufatura pode ser continuamente analisado em aspectos de sua distribuição física e requisitos de produção. No caso de surgirem alterações necessárias ao melhor desempenho do sistema, o Ciclo de Desenvolvimento é reiniciado, aproveitando-se dos modelos armazenados em bibliotecas.

A seguir, serão apresentados em detalhes o AD e sua aplicação a cada etapa do Ciclo de Desenvolvimento. Ao final do capítulo, serão apresentadas considerações sobre a aplicação de HiL e a aplicação da proposta de Busetti e Santos (2006).

3.1 Ambiente Dinâmico

Para permitir que a implementação em três etapas seja possível, uma plataforma computacional deve ser concebida de forma a estabelecer uma comunicação entre o CLP Central e a Plataforma de Simulação. Tanto o Ambiente Dinâmico (AD), como a Plataforma de Simulação, podem ser implementados em um mesmo computador.

No AD, é desenvolvida uma Biblioteca de Modelos de Comunicação (BMC), ilustrada na Figura 14, na qual, cada modelo possui um canal de comunicação para cada comando ($Cmd\alpha_i$, com $i = 1, 2,..., n$.), e, outro, para cada resposta ($Rsp\beta_i$), relacionados a eventos controláveis e não-controláveis, respectivamente. Observa-se que, para cada comando e resposta, há sempre um reconhecimento de recebimento, respectivamente, $AckCmd\alpha_i$ e $AckRsp\beta_i$. Isto é feito para informar ao CLP Central que a Plataforma de Simulação recebeu o comando, informação necessária para que a implementação da metodologia de Vieira (2007) em CLP seja possível. Da mesma forma, mas em sentido inverso, o CLP Central informa a Plataforma de Simulação sobre o reconhecimento da resposta gerada. A geração desses reconhecimentos é tratada em Diogo *et al* (2008), e maiores detalhes serão dados nas seções 3.3, 4.2 e 4.3.

Adicionalmente, o AD possui uma interface com o usuário. O principal objetivo dessa interface é informar se o subsistema físico está somente em modo de simulação (*offline*), ou se a simulação deve ocorrer paralelamente à execução do subsistema físico real (*online*), ou seja, quando algum CLP Local está conectado ao CLP Central através de Tecnologias de Controle e Comunicação (TCC). A escolha por modo *offline* ou *online* deve ser tomada, pois o CLP Central só pode receber respostas da Plataforma de Simulação se a arquitetura de controle supervisório (ACS) estiver sob processo de validação. Quando o CLP Central está conectado a algum CLP Local, o Procedimento Operacional Real (POR) deste irá gerar as respostas necessárias; portanto, o modelo na Plataforma de Simulação deve ser impedido de gerar tais respostas, e, sim, recebê-las, com o intuito de tomar o tempo de execução da operação no processo produtivo, assunto tratado na seção 3.2.

Na Figura 14, é ilustrado somente como o AD integra o CLP Central com a Plataforma de Simulação, a integração com algum CLP Local não é tratada. Observa-se que os níveis Supervisores Modulares (SM) e Sistema-Produto (SP)

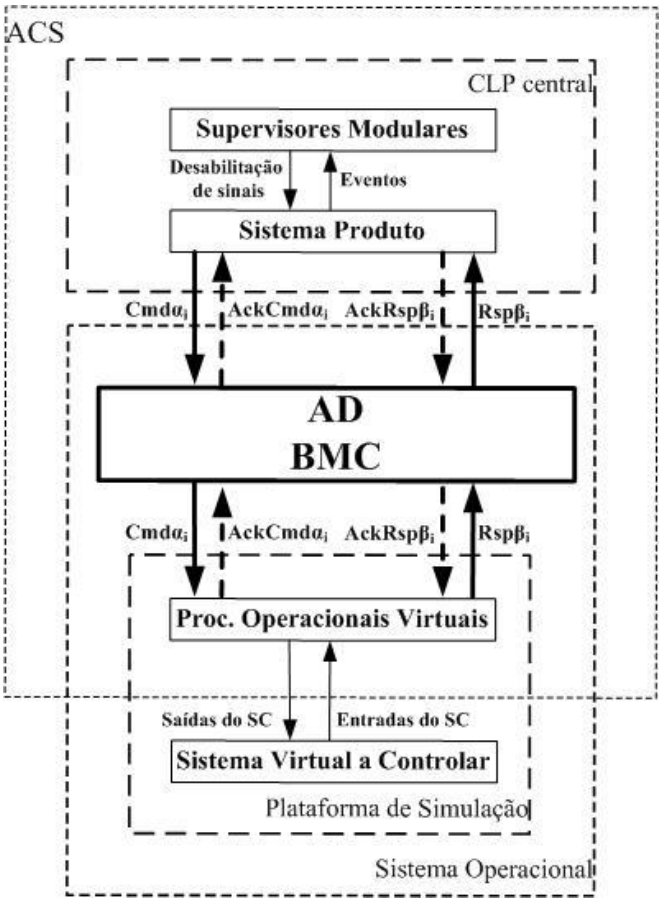

Fig. 14. AD e BMC como integração do CLP central com a plataforma de simulação.

estão implementados no CLP Central, enquanto que os níveis Procedimentos Operacionais Virtuais (POV) e Sistema Virtual a Controlar (SVC) estão implementados na Plataforma de Simulação. A Plataforma de Simulação pode ser desenvolvida em um *software* de simulação (ex.: Arena) e este é executado em um sistema operacional para computadores (ex.: Windows). O AD e a BMC encontram-se como integração do CLP Central com a Plataforma de Simulação. O AD e a BMC são implementados em um sistema de supervisão (Ex.: Elipse E3), devido à facilidade de geração de bibliotecas neste tipo de *software*. A comunicação entre o CLP Central e a Plataforma de simulação se dá pela leitura e escrita de comandos e respostas em Base de Dados (BD) implementada no AD.

Através de tempos de varredura programados na Plataforma de Simulação e no AD, a leitura e escrita de comandos e respostas é realizada constantemente. Isto é, o estado de uma seção de memória interna do CLP Central é lido pelo AD em intervalos de tempo programados, particularmente o estado das variáveis referentes a comandos. Já o estado das respostas na BD é escrito constantemente na região de memória reservada no CLP Central. Isto pode ser um fator limitante da proposta deste trabalho, pois pode ocorrer alteração de eventos em tempo menor que o programado na varredura dos *softwares*. Tais eventos podem não ser percebidos pelo CLP Central, ocasionando bloqueio na evolução de eventos da ACS.

Observa-se, ainda, na Figura 14, que uma TCC foi inserida para que os dados trafeguem pelo AD, integrando o CLP Central e a Plataforma de Simulação. Esta TCC é uma BD que foi implementada no AD para executar a comunicação entre o CLP Central e a Plataforma de Simulação. Adicionalmente, *drivers* de comunicação são inseridos no AD, para permitir a comunicação do CLP Central com o AD e deste com a Plataforma de Simulação. *Drivers* são programas que permitem a comunicação de aplicações com periféricos e também com sistemas de BD. Observa-se que o AD e a Plataforma de Simulação podem ser implementados em uma mesma plataforma computacional, bastando que uma tabela do BD execute a comunicação entre eles.

A Figura 15 exalta onde a proposta deste trabalho atua sobre o Ciclo de Desenvolvimento de Busetti e Santos (2006): a inserção do AD na ACS de Queiroz e Cury (2002) aplicado aos três passos da etapa de implementação. Observa-se que os modelos gerados na etapa de modelagem são utilizados para

formar a BMC. Ou seja, os modelos de subsistemas são analisados para se verificarem os comandos e respostas, a fim de se construírem os modelos de comunicação.

Primeiramente, considera-se somente o passo de Simulação e observa-se que a ACS está implementada em um CLP Central. Através do AD, a comunicação entre a Plataforma de Simulação e o CLP Central é realizada. Os POV executam as operações sobre o SVC. Observa-se, também, que é a Plataforma Simulação que gera a resposta.

Já no passo de Simulação + TCC, um CLP $Local_n$ é inserido ao sistema, portanto, um subsistema real está conectado. Neste caso, as respostas do subsistema físico são enviadas ao CLP Central e à Plataforma de Simulação (setas com duplo sentido). A resposta enviada à simulação tem o intuito de sincronizá-la. As respostas geradas pelos subsistemas reais, ainda em simulação, são enviadas ao CLP Central, para que a ACS evolua corretamente.

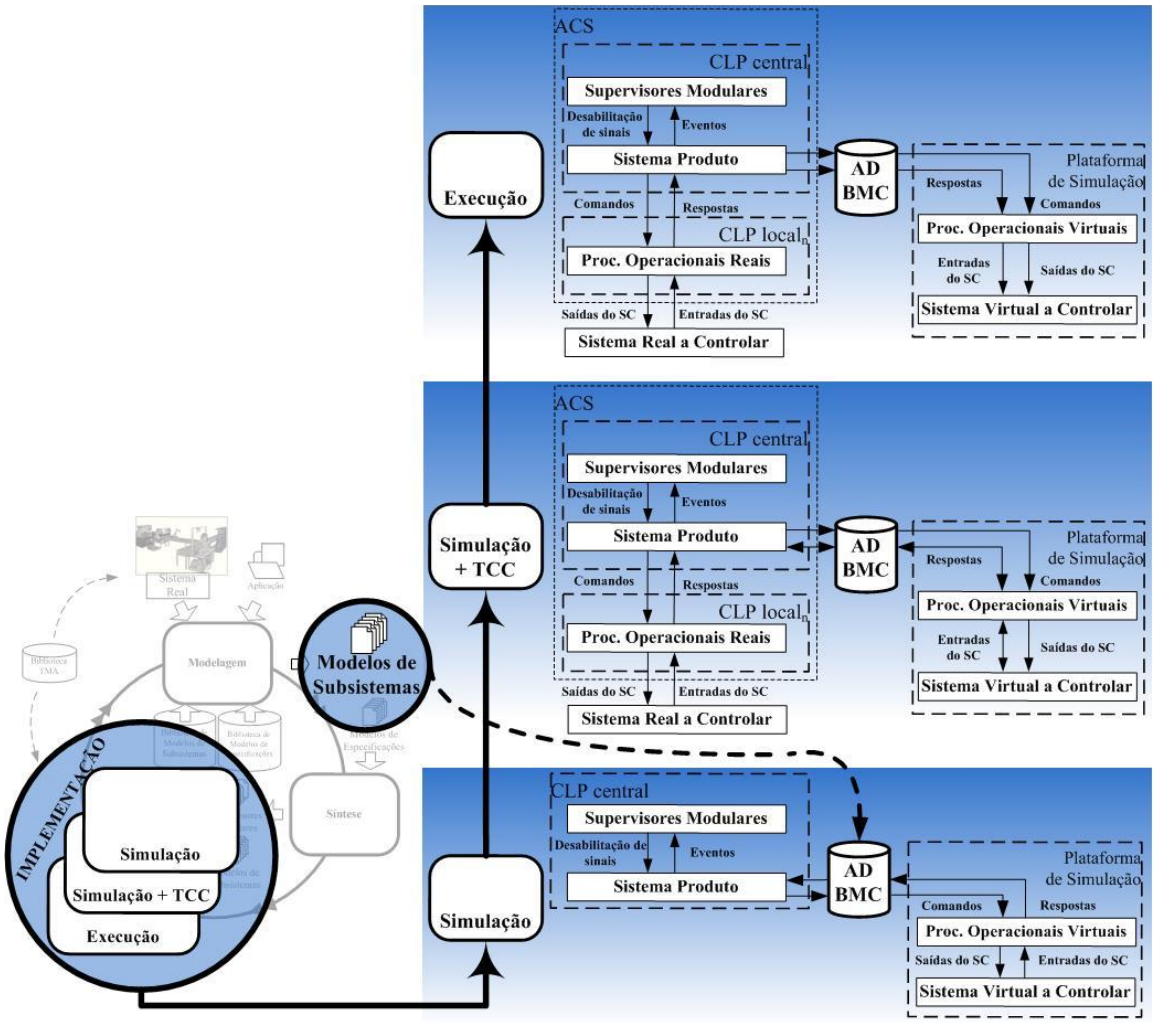

Fig. 15. AD aplicado aos três passos da etapa de implementação.

No último passo, a execução, todos os *n* CLP Locais foram inseridos. Neste passo, não há mais respostas geradas pela simulação, todas são geradas pelos CLP Locais (inversão das setas das respostas). Como no passo anterior, as respostas são enviadas para a simulação, a fim de haver sincronismo na mesma.

A seguir, é descrita uma série de operações que o AD deve executar no modo de simulação:

i) A leitura de comandos ($Cmd\alpha_i$, com $i = 1, 2, ..., n$) gerados no CLP Central e a escrita na plataforma de simulação. Isto é feito para iniciar o POV correspondente na Plataforma de Simulação.

ii) A recepção de notificações de reconhecimento de comandos ($AckCmd\alpha_i$), escritas pela Plataforma de Simulação. Isto é feito para desativar o comando correspondente no CLP Central, uma vez que o POV já tenha sido iniciado.

iii) A leitura das respostas ($Rps\beta_i$) geradas na Plataforma de Simulação e a escrita no CLP Central, a fim de gerar o evento não-controlável correspondente, uma vez que o POV tenha terminado sua tarefa. Isto permitirá a atualização dos SM, estabelecendo um novo conjunto de desabilitações de eventos controláveis e um novo conjunto de habilitações de outros eventos controláveis, permitindo, assim, a evolução dos modelos no SP.

iv) A leitura de notificações de reconhecimentos de respostas ($AckRps\beta_i$) geradas pelo CLP Central e consequente escrita na Plataforma de Simulação. Isto é feito para desativar, na Plataforma de Simulação, a resposta correspondente, uma vez que o CLP Central tenha atualizado o evento correspondente.

Entretanto, deve haver uma interface para alternar entre o modo de Simulação e o modo de Ambiente Real, afetando os itens *i* e *iv*. No modo de Simulação, o AD coleta as respostas da Plataforma de Simulação e as envia ao CLP Central, como já conhecido. Escolhendo o modo de Ambiente Real, o AD deve inverter o sentido das respostas e das notificações de reconhecimento de respostas quando um subsistema real está conectado. Isto é feito para impedir a geração de

respostas pela Plataforma de Simulação. Neste caso, a Plataforma de Simulação é informada pelo AD sobre o modo de operação, e, então, recebe as repostas correspondentes do POR e gera a notificação de reconhecimento ao CLP Local. Desta forma, é possível adquirir dados reais do sistema de manufatura.

A justificativa da necessidade da BMC se deve ao fato de que o projetista pode reaproveitar modelos para construir outros, economizando, assim, tempo na execução de projetos de sistemas de manufatura. Particularmente, este recurso permite armazenar modelos de subsistemas (e seus respectivos sinais, de acordo com o modelo de implementação de três níveis), formando a BMC. Isto permite certa flexibilidade ao projetista.

3.2 Etapa de Simulação

De acordo com o Ciclo de Desenvolvimento proposto por Busetti e Santos (2006), a primeira etapa deste ciclo corresponde à execução da simulação da planta em estudo. Conforme descrito no Capítulo 2, a simulação ocorre para validar a Arquitetura de Controle Supervisório (ACS). Para tornar possível esta abordagem, o Ambiente Dinâmico (AD) está conectado à Plataforma de Simulação, de modo a que eles leiam e escrevam eventos entre si. Neste caso, o objetivo do AD é permitir que a simulação ocorra pela leitura de comandos do CLP Central. Além disso, é necessário conceber uma codificação na Plataforma de Simulação (interface na Figura 16), de modo que a comunicação entre o AD e a simulação seja orientada com base na leitura dos comandos gerados a partir do CLP Central e escritos em uma Base de Dados (BD).

Com base em uma arquitetura dividida em níveis, conforme proposto pela ISO TR10314-1 (1990), uma adaptação do modelo proposto por Hibino *et al*. (2006) foi inserida no modelo da ACS de Queiroz *et al* (2001). O modelo de Hibino *et al* (2006) consiste de um ambiente para se conectarem fábricas reais e virtuais, através de simulação distribuída do sistema de manufatura real. Desta forma, a simulação ficou dividida em três níveis, de acordo com a Figura 17 (Diogo *et al*, 2008).

O Nível de Interface é responsável pela comunicação entre a plataforma de simulação e o AD. Seu principal objetivo é realizar a leitura dos comandos enviados pelo AD, processá-los e executar o POV relacionado a cada

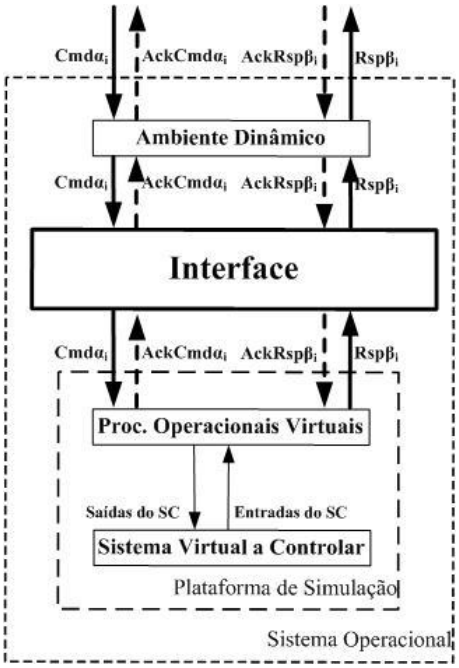

Fig. 16. Interface para permitir a comunicação entre o AD e a Plataforma de Simulação.

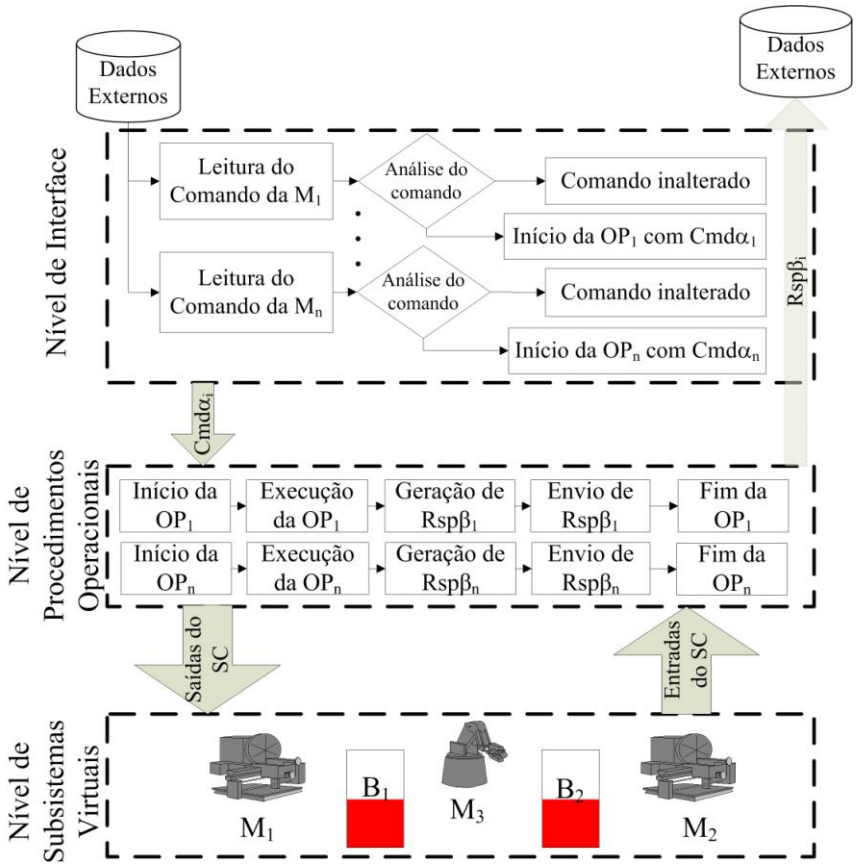

Fig. 17. Plataforma de simulação (Diogo *et al*, 2008), com *i = 1, 2, .., n*.

comando. Os comandos enviados a partir do CLP Central ao AD são armazenados em uma tabela da BD, e lidos pela Plataforma de Simulação. Nesta plataforma, uma estrutura de tomada de decisão é definida de acordo com um determinado valor lógico. O Nível de Interface envia um sinal para o Nível de Procedimentos Operacionais Virtuais, caso contrário, ele espera por uma nova leitura ao BD.

O Nível de Procedimentos Operacionais Virtuais é responsável pelo envio de comandos ao Nível de Subsistemas Virtuais. A fim de minimizar o tempo de envio dos dados, o evento correspondente a uma resposta será enviado do Nível de Procedimentos Operacionais Virtuais ao AD, sem serem tratados pelo Nível de Interface. Fica a cargo do Nível de Procedimentos Operacionais Virtuais a geração de registros, que deverão conter dados quantitativos sobre a planta virtual e sua capacidade produtiva. Assim, os registros, além de possibilitarem a validação da ACS, serão úteis para a tomada de decisão sobre os recursos disponíveis para o planejamento da produção, pois, através da tomada de tempo registrada, será possível se avaliar se o sistema de manufatura supre a demanda de produção exigida.

O Nível de Subsistemas Virtuais é responsável pela simulação dos subsistemas que compõem a planta em estudo. Nesse nível, há comunicação direta com o Nível de Procedimentos Operacionais Virtuais e troca de comandos e respostas com o mesmo. Também nesse nível, os modelos virtuais são implementados levando-se em conta a disposição física do sistema de manufatura real. Dessa forma, para cada modelo de subsistema real há um subsistema virtual correspondente.

De acordo com a etapa de implementação proposta, os POV podem ser simulados um a um, como ilustrados na Figura 18. Primeiramente, somente um modelo virtual é simulado; depois, novos modelos são progressivamente agregados, até se atingir a simulação completa do sistema. Ao se fazer isso, é possível se determinar, gradativamente, se a ação de controle imposta pelo CLP apresenta erros, ou se o projetista conseguiu atingir as completudes. A inserção progressiva dos POV permite verificação sistemática da síntese da ACS, bem como, da confiabilidade na inserção dos subsistemas. Observa-se, ainda, que essa inserção gradativa é uma vantagem adicionada à etapa de Simulação da metodologia de Busetti e Santos (2006), onde todos os subsistemas são simulados ao mesmo tempo.

3.3 Etapa de Simulação + TCC

De acordo com Bullock *et al*. (2004), em um sistema de simulação *off-line* as operações são executadas o mais rapidamente possível. Dependendo do sistema operacional e do sistema computacional usado, é possível se simularem horas, dias, semanas e assim por diante em poucos segundos. Isto é feito para diminuir o tempo de avaliação e testes dos projetos de sistemas de manufatura. Já, se um subsistema físico está conectado, ele é executado em um tempo permitido pela sua configuração física. Entretanto, a parte do sistema que ainda está sob simulação é executada o mais rapidamente possível.

Gradualmente, os subsistemas físicos reais são inseridos, como ilustrado na Figura 19. Um subsistema físico real pode ser considerado como uma máquina conectada a um CLP Local que executa operações do sistema de manufatura. Cada subsistema físico real é executado por um Procedimento Operacional Real (POR) relacionado. A troca de sinais entre o CLP Central (onde estão implementados os níveis SM e SP da ACS) e o CLP Local (onde está implementado um determinado POR) é realizada pela escrita de comandos e respostas na memória interna de tais CLP. Dessa forma, os subsistemas físicos

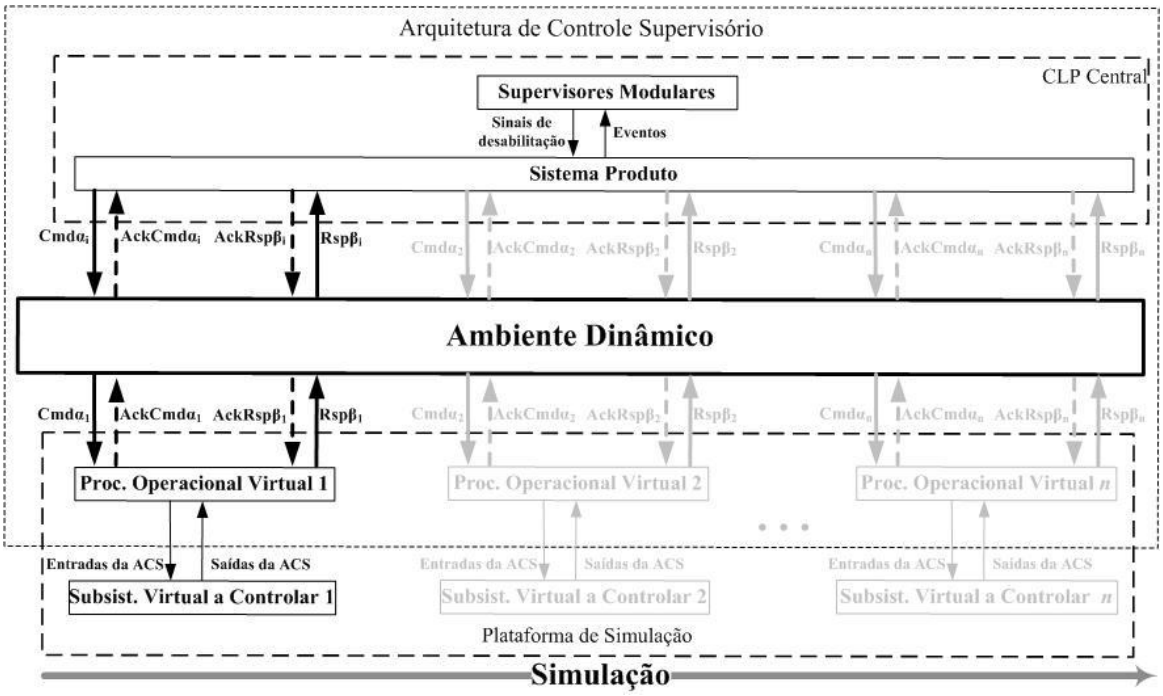

Fig. 18. O AD inserido no passo de simulação ($i = 1, 2, ..., n$).

reais são executados gradativa e paralelamente à simulação dos subsistemas físicos virtuais. Durante essa etapa, o AD integra os CLP (Central e Local) e Plataforma de Simulação.

A resposta originada na Plataforma de Simulação, resultante de um evento não-controlável de um subsistema físico virtual, deve ser enviada à ACS somente se a mesma estiver sob validação, como ilustrado na Figura 18. No caso de uma simulação paralela a um subsistema físico real, este gera uma resposta depois do fim da operação relacionada, que é enviada ao CLP Central e à Plataforma de Simulação, como demonstrado na Figura 19.

A resposta deve ser enviada à Plataforma de Simulação, com o objetivo de obter o tempo entre o início e fim do ciclo de um subsistema. Dessa forma, é possível obter-se o tempo aproximado das operações do sistema de manufatura que estão sendo executadas. Os dados armazenados podem informar se o sistema de manufatura atende às exigências de produção. Contudo, esta sincronização pode não ocorrer em tempo real, dependendo do sistema operacional onde o AD será implementado (exemplo: Unix e Windows têm diferenças relevantes quanto a tempo real).

A implementação de todos os CLP Locais (e subsistemas físicos associados) permite que o comportamento global da planta de manufatura seja

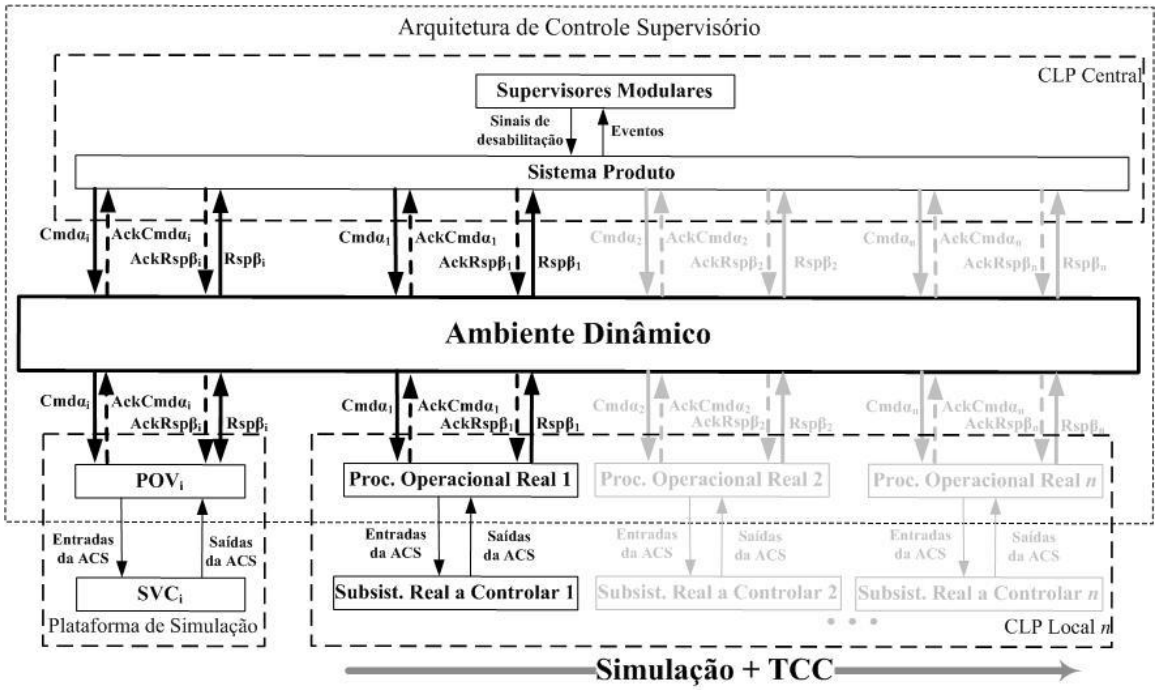

Fig. 19. O AD inserido no passo de simulação + TCC e no passo de execução ($i = 1, 2, ..., n$).

avaliado. Nesta etapa, é possível validar-se progressivamente a ACS (*software*) quando está conectada ao sistema físico, e, também, analisar questões relacionadas com a distribuição do sistema físico (*hardware*). Também é possível capturar dados do sistema físico através da Plataforma de Simulação. Dessa forma, é possível se determinarem as necessidades e capacidades da produção, e, consequentemente, verificar se o sistema de manufatura está adequado para suportar a demanda de produção.

3.4 Etapa de Execução

Nesta etapa, todos os subsistemas físicos reais estão conectados ao sistema de manufatura e o comportamento global pode ser completamente analisado. Isto significa que todos os dispositivos (ex.: CLP, etc) que têm os POR estão conectados aos respectivos sensores e atuadores dos subsistemas físicos. As respostas geradas pelos CLP Locais são enviadas ao CLP Central e para a Plataforma de Simulação (sincronização da simulação). Assim, as respostas enviadas à ACS não passam pelo AD, somente as respostas enviadas para a plataforma de simulação, como ilustrado na Figura 15.

Ainda nesta etapa, os processos produtivos podem ser continuamente analisados na Plataforma de Simulação, pela aquisição dos tempos de produção através do Ambiente Dinâmico (AD). De acordo com esses dados, novas capacidades e necessidades produtivas podem ser determinadas. Se isto ocorrer, todo o Ciclo de Desenvolvimento é reiniciado, mas somente para determinar os novos modelos que representam o comportamento dos subsistemas e as novas especificações comportamentais. Depois da síntese de supervisores, a etapa de Implementação é executada novamente.

O reinício do Ciclo de Desenvolvimento pode ocorrer se uma reconfiguração do sistema é necessária quando da inserção de novos produtos a serem fabricados pelo sistema de manufatura. Esta reconfiguração pode ocorrer por modificações nas operações ou, até mesmo, pela inclusão ou retirada de máquinas do sistema de manufatura.

3.5 Conclusão do capítulo

Este capítulo apresentou o modo como as metodologias do Capitulo 2 estão sendo aplicadas na proposta da metodologia deste trabalho e como elas se adaptaram a ele. Novas técnicas foram definidas para facilitar o desenvolvimento dos projetos de sistemas de manufatura. Adicionalmente, podem ser vistas as tecnologias mínimas que devem fazer parte desta metodologia: controladores industriais, redes industriais, banco de dados, *softwares* de supervisão e também de simulação.

Mas, outros pontos ainda devem ser levados em consideração. É importante salientar que os conceitos de *Hardware-in-the-Loop* (HiL) são utilizados na proposta de desenvolvimento do Ambiente Dinâmico, porém, existe limitação na utilização de sistemas operacionais que não sejam adequados a aplicações de simulação em tempo real, bem como à potencialidade do *hardware* utilizado.

Observa-se, também, que, ao longo da etapa de Implementação, os subsistemas físicos virtuais são gradativamente substituídos pelos reais; porém, na proposta do AD, os subsistemas virtuais são mantidos quando são inseridos os reais. Então, é possível se acompanhar, virtual e paralelamente, a evolução do subsistema físico (virtual e real).

4 VALIDAÇÃO DO AMBIENTE DINÂMICO

Este capítulo demonstra uma aplicação como exemplo para validação do Ambiente Dinâmico (AD). Primeiramente, um exemplo de sistema da manufatura composto por duas máquinas é apresentado, conforme a Figura 20. A modelagem do comportamento livre das máquinas e das especificações comportamentais é feita na primeira etapa do Ciclo de Desenvolvimento proposto por Busetti e Santos (2006). Posteriormente, a síntese dos supervisores é executada na segunda etapa. Finalmente, a implementação em três passos é realizada.

Para que a validação do AD fosse possível, os modelos dos subsistemas e das especificações obtidos na primeira etapa do Ciclo de Desenvolvimento foram traduzidos para linguagem compatível com CLP, assim como os supervisores sintetizados na segunda etapa. Isto é feito com o suporte do método proposto por Vieira (2007), que consiste na tradução dos autômatos em SFC (*Sequential Function Chart*). Porém, alguns CLP não possuem tal linguagem; então, os SFC correspondentes são representados em linguagens de mais baixo nível, como a *Ladder Diagram*. Adicionalmente, uma Plataforma de Simulação para as duas máquinas foi desenvolvida, conforme apresentado em Diogo *et al* (2008). Essas implementações foram realizadas para que o AD execute sua principal função, que é ler e escrever comandos na Plataforma de Simulação e no CLP Central.

Ao final da implementação, uma mudança no sistema de manufatura é proposta, com a inserção de uma nova máquina. Isto visa à verificação da flexibilidade do AD. Dessa forma, é possível ver-se que os sistemas de manufatura atuais podem sofrer mudanças quando da alteração do produto fabricado ou por novas exigências de demanda.

A proposta desta dissertação está aplicada na última etapa do Ciclo de Desenvolvimento, portanto, a validação do AD ocorre nela. Para isso, o exemplo de manufatura é descrito a seguir, e, posteriormente, expandido.

4.1 Exemplo de Sistema de Manufatura

Esta seção apresenta um sistema de manufatura hipotético, que será empregado para ilustrar o ambiente de implementação proposto. As operações foram realizadas manualmente em CLP, para facilitar o estudo. A Figura 20 apresenta o sistema de manufatura, que é composto por duas máquinas e um *buffer* entre elas. Ambas as máquinas são centros de usinagem. Os autômatos associados às máquinas também são ilustrados na Figura 20 e são modelados da forma mais simples possível, ou seja, um estado do autômato representa máquina em repouso, e, o outro, máquina em operação. Portanto, G_1 no estado 0 significa que a máquina M_1 está em repouso, e, no estado 1, em operação. O mesmo ocorre para G_2 e M_2. Para esse sistema de manufatura, os eventos α_1 e α_2 das máquinas M_1 e M_2 são eventos controláveis (início de operação), e β_1 e β_2, não-controláveis (fim de operação). Também, considera-se neste caso que a peça processada por M_1 é automaticamente transportada para a M_2.

O objetivo, neste exemplo, é usar de forma otimizada os recursos disponíveis, de maneira correta e segura, ou seja, o processamento de várias peças no sistema não pode colocar mais de uma peça no *buffer* B_1 ou tentar retirar peça

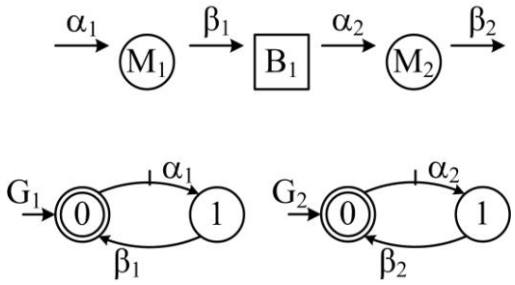

Fig. 20. Sistema de manufatura e respectivos modelos (autômatos).

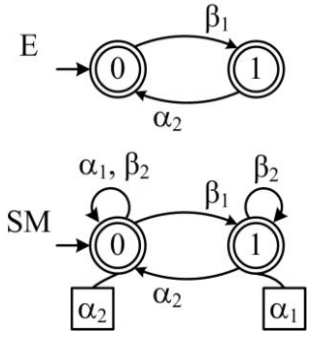

Fig. 21. Especificação para o *buffer* entre M_1 e M_2 e supervisor reduzido resultante.

quando não há nenhuma a ser retirada. A especificação que atende a essa restrição é apresentada na Figura 21. Essa figura também apresenta o supervisor reduzido, obtido pelo procedimento de síntese. Os eventos nos quadrados são desabilitados quando o estado respectivo do SM está ativo.

A implementação no CLP Central do Supervisor Modular e do Sistema Produto que comporta os modelos de operação é apresentada a seguir. Essa implementação segue o método de implementação proposto por Vieira (2007).

4.2 Implementação dos modelos no CLP Central

Vieira (2007) propõe um método para implementação em CLP da Arquitetura de Controle Supervisório (ACS) proposta por Queiroz *et AL* (2001). Tal modelo possui um programa principal, implementado em SFC, em que cada passo determina o modo de operação do sistema. Este programa foi simplificado neste trabalho, pois o objetivo é validar o Ambiente Dinâmico (AD). Os passos referentes à operação manual e estado de emergência foram retirados da proposta original de Vieira (2007). Isto é ilustrado na Figura 22.

A ação *Action SI*, associada ao passo *SI* (*Software Inicialization*), que foi ativado pela variável *Sinit*, é responsável pela inicialização do SFC referente ao

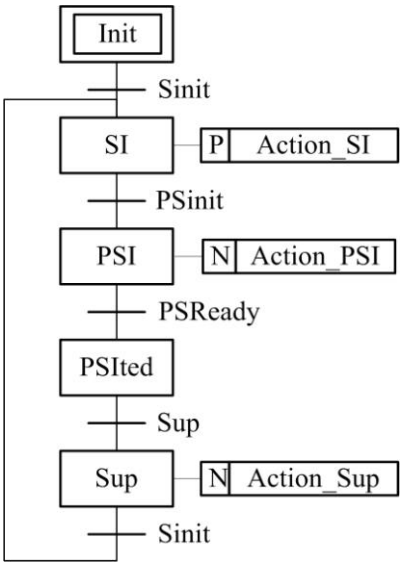

Fig. 22. SFC Main implementado no CLP Central da proposta do AD.

Supervisor Modular (SM), aos modelos que compõem o Sistema-Produto (SP) e aos Procedimentos Operacionais Reais (POR) implementados nos CLP Locais. Os POR também são implementados em SFC.

Considera-se que a Plataforma de Simulação, na qual os Procedimentos Operacionais Virtuais (POV) estão implementados, já apresenta todas as variáveis com os valores iniciais, quando iniciada. Dessa forma, não é necessário executar-se uma comunicação através do AD para a inicialização de variáveis na simulação. Considera-se, ainda, que *Action_SI* está implementado em um FB do CLP Central e tem a função de iniciar as variáveis daquele CLP, assim como as variáveis dos CLP Locais.

Considerando-se que todas as variáveis do sistema estejam com seus valores iniciais, a variável *PSinit* é ativada, tornando possível a ativação do passo *PSI* (*Physical System Inicialization*). A esse passo está associada a ação *Action_PSI*, responsável por conduzir todo o sistema físico ao estado inicial. A inicialização física só acontece quando subsistemas físicos reais estão conectados ao CLP Central, pois a Plataforma de Simulação e o AD não prevêem a inicialização dos subsistemas físicos virtuais. Quando todo o sistema físico estiver iniciado, a variável *PSready* torna-se ativa, ativando o passo *PSIted*.

Neste ponto, o sistema aguarda que o usuário o coloque em modo de operação supervisionado. Isto é feito quando o usuário torna a variável *Superv* verdadeira. O sistema é, então, coordenado pelo conjunto de supervisores, através da desabilitação de sinais ditada pela ação de controle. A implementação do FB que comporta a ação de supervisão *action_sup* é tratada em detalhes mais adiante.

O CLP Central, utilizado nesta implementação, é um micro CLP que não suporta linguagem SFC; portanto, o SFC *Main* foi traduzido para a linguagem *Ladder Diagram* ,através de procedimentos formais.

4.2.1 Implementação dos FB referentes aos modelos do SP

O Sistema-Produto (SP) do exemplo proposto é formado por dois autômatos, G_1 e G_2 conforme Figura 20. Ambos foram traduzidos para SFC pelo

método de Vieira (2007), e cada um é implementado em um FB (*G1* e *G2*). Esses modelos de SFC obtidos são ilustrados na Figura 23. Como o CLP utilizado para implementação não suporta linguagem SFC, os SFC obtidos são traduzidos para linguagem *Ladder Diagram*. A seção de código referente ao SFC de G_2 é ilustrada na Figura 24 (A seção de código de G_1 é semelhante à de G_2).

Observa-se, na Figura 24, a existência de uma variável negada denominada *G2_EVT* nas transições para os passos. Essa variável não é observada nas transições dos SFC. Ela foi inserida para se evitar o efeito avalanche. Esta variável é ativada para informar que houve evolução do modelo e sua desativação é

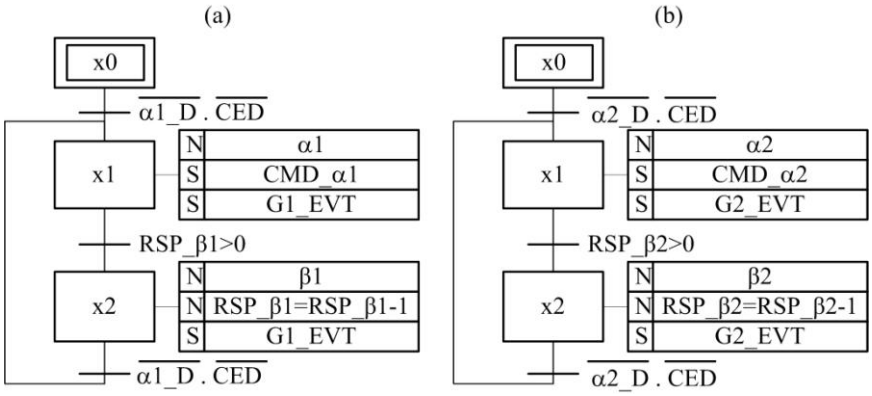

Fig. 23. a) SFC referente ao autômato G_1; b) SFC referente ao autômato G_2.

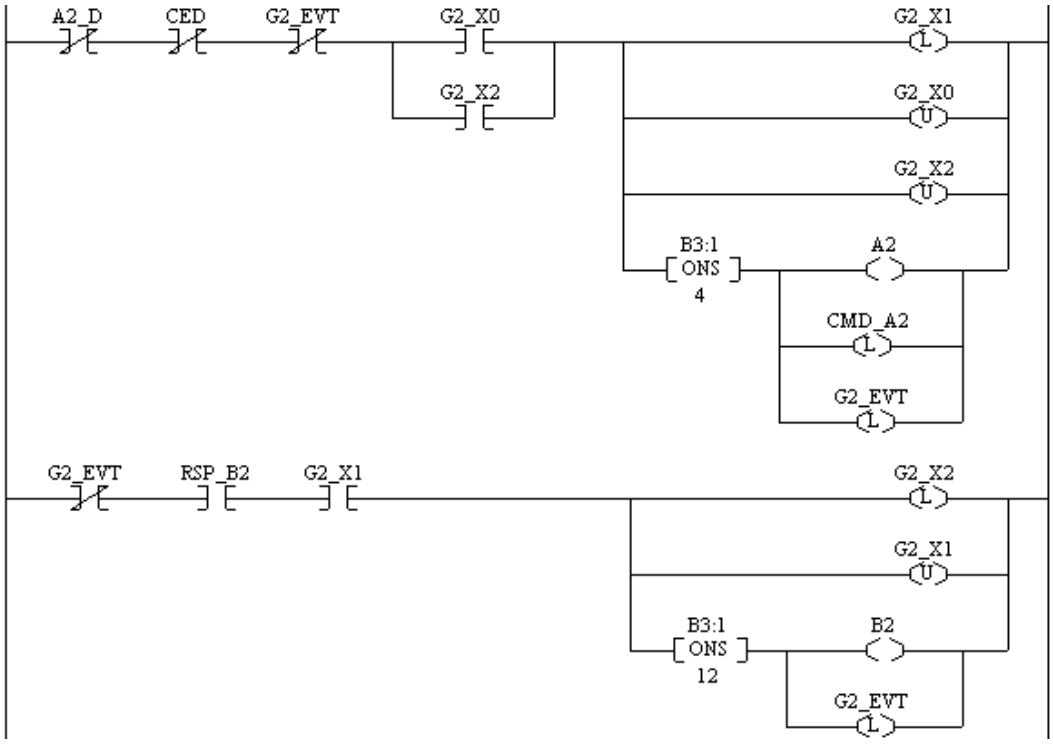

Fig. 24. Seção de código *Ladder* referente ao SFC de G_2.

tratada na seção 4.2.6.

Também se observa na Figura 24 que a transição *RSP_B2* na seção de código, referente à *RSP_β2* no SFC, não está na forma de comparação (se maior do que 0, na figura 23). Como os modelos obtidos não geram sucessivas respostas, não foi preciso executar tal comparação na implementação. Dessa forma, a seção de código torna-se mais simples.

Ainda há comentários sobre algumas variáveis. A variável *CED* foi introduzida para suspender todo evento controlável para priorizar o tratamento de eventos não-controláveis. A seção de código dessa variável será tratada na seção 4.2.6. As variáveis associadas às funções lógicas *ONS* (*One-Shot Rising* – detecção de borda de subida em CLP da linha Allen-Bradley, MicroLogix 1200 com programador RSLogix 500) são inseridas, para que as variáveis posteriores somente sejam ativadas em apenas um ciclo de varredura do CLP. E, finalmente, *A2_D* é uma variável ativada pelo estado inicial do supervisor, que desabilita a ocorrência do evento α_2. A seção posterior trata da seção de código que ativa e desativa esta variável.

Como exemplo, analisando a primeira linha da seção de código da Figura 24, é possível se observar que, se as variáveis *A2_D, CED* e *G2_EVT* não estiverem ativas e o SFC de G_2 estiver no passo *x0* ou *x2*, então, o passo *x1* do referido SFC é ativado com retenção (*G1_X1*). Nesta mesma linha, é desativado o passo anterior (*x0* ou *x2*). Adicionalmente, é ativada, com retenção, a sinalização de evolução do modelo (*G2_EVT*), é gerado apenas uma vez o evento α_2 (*A2*) e é ativado, com retenção, o respectivo comando, *CMD_A2*, que é lido pelo AD e escrito na Plataforma de Simulação a fim de iniciar o Procedimento Operacional Virtual (POV) correspondente. A desativação deste comando é realizada pelo FB no qual a ação de supervisão do SFC *Main* está implementada. Para a segunda linha da seção de código, faz-se a analogia lógica realizada para a primeira linha.

4.2.2 Suspensão de tratamento de eventos através dos FB associados aos modelos do SP

O exemplo tratado possui apenas um supervisor, que é compartilhado pelos dois modelos (G_1 e G_2). Sempre que um evento é tratado em um dos modelos, e este compartilha uma célula de controle com outro modelo, deve ocorrer a suspensão do tratamento de eventos e ela deve ser mantida até que todo estado ativo do supervisor tenha sido atualizado (Vieira, 2007). Para cada FB que representa um modelo do SP, é associado outro FB responsável pela suspensão de eventos. No caso do exemplo de duas máquinas, há dois FB que realizam a suspensão: *DG1 e DG2*. As seções de código implementadas nesses FB são ilustradas nas Figuras 25a e 25b, respectivamente, nos quais, a sinalização de evolução de um modelo ou outro ativa a suspensão de tratamento de eventos em G_1, através de *G1_D*, e em G_2, através de *G2_D*.

4.2.3 Implementação do FB referente ao SM

O autômato obtido como supervisor do sistema a ser controlado, ilustrado na Figura 21, é traduzido para um SFC. O SFC obtido é ilustrado na Figura 26. Da mesma forma que foi feito para os SFC do SP, este SFC deve ser traduzido em linguagem *Ladder Diagram* e é implementado em um novo FB.

A seção de código que representa este SFC em *Ladder Diagram* é ilustrada na Figura 27. Analisando essa programação, observa-se que, se o estado

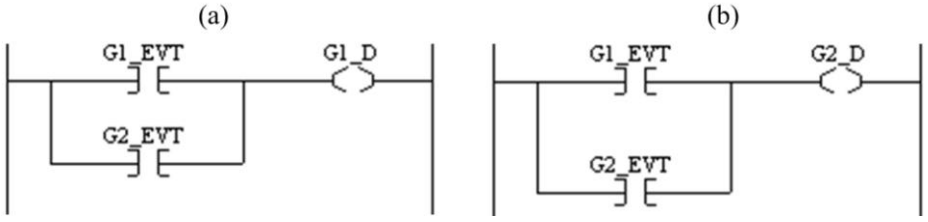

Fig. 25. a) Seção de código referente à suspensão do tratamento de eventos em *G1*; b) Seção de código referente à suspensão do tratamento de eventos em *G2*.

ativo do supervisor é *x1* (*S1_X1*) e ocorre o evento $\alpha 2$ (variável *A2* está ativada), então, o estado do supervisor passa a ser *x0* (*S1_X0*). Já, se ocorrer o evento $\beta 1$ (variável *B1* está ativada) e o estado *x0* do supervisor estiver ativo, então, o estado ativo do supervisor passa a ser *x1* (*S1_X1*). Observa-se que cada estado do supervisor tem uma ação de controle, desativando eventos controláveis. Isto é visto através da sinalização de $\alpha 2\_D\_S1$, no passo *x0* do SFC, e de $\alpha 1\_D\_S1$, no passo *x1*. Também é possível se observar como tudo foi programado, nas duas últimas linhas da seção de código da Figura 27.

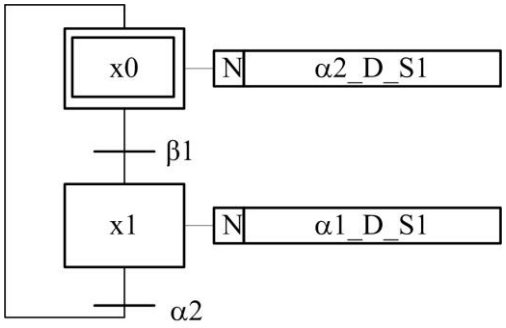

Fig. 26. SFC referente ao supervisor SM.

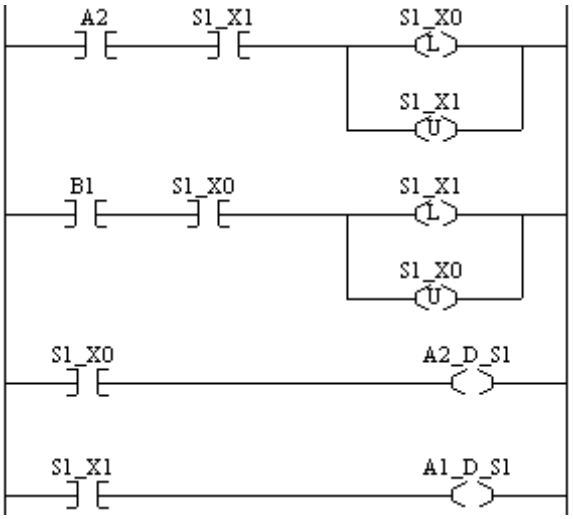

Fig. 27. Seção de código referente ao supervisor SM.

4.2.4 Implementação do FB *MS*

A ação *Action_Sup,* no SFC *Main*, realiza a chamada de alguns FB. Um deles, é o FB *MS*, que, por sua vez, realiza, a cada ciclo de varredura do CLP, a chamada sequencial dos FB referentes aos SM (Vieira, 2007). No exemplo proposto, há somente um supervisor, mas, ainda assim, é implementado o *MS*, pois, no caso de modificações do programa, já haverá uma estruturação do programa como um todo. Outra função do FB *MS* é a ativação dos sinais de desabilitação da ação de controle do supervisor que impedem a evolução dos modelos de subsistemas. Para o exemplo inicial deste capítulo, tem-se a seção de código representada pela Figura 28, sendo que a primeira linha faz a chamada cíclica do FB que compõe o SM. *A1_D_S1* e *A2_D_S1* são variáveis definidas da implementação da Figura 27. Essas variáveis, respectivamente, determinam a desabilitação de eventos controláveis, através de *A1_D* e *A2_D*.

4.2.5 Implementação do FB *PS*

O FB *PS* é chamado pelo programa que comporta o SFC *Main* através da ação de supervisão *Action_Sup*. Por sua vez, o FB *PS* realiza a chamada sequencial de todos os FB referentes aos modelos de subsistemas que compõem o nível SP da ACS de Queiroz *et al* (2001). A seção de código que representa esse FB está ilustrada na Figura 29. Observa-se que, antes de ser chamado o FB correspondente a um modelo de subsistema, é realizada a chamada do respectivo

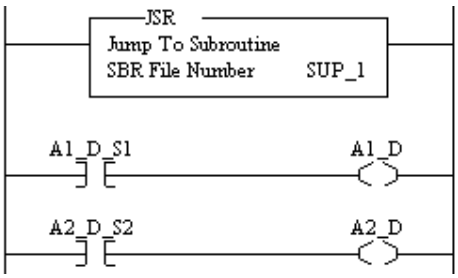

Fig. 28. Seção de código referente ao supervisor SM.

FB de suspensão de tratamento de eventos. O FB *PS* só irá realizar a chamada dos FB *G1* e *G2* se a suspensão de tratamento de eventos em *DG1* e *DG2* não estiver ativa, ou seja, se as variáveis *G1_D* e *G2_D* não estiverem ativadas.

4.2.6 Implementação do FB *Action_Sup*

Uma parte da seção de código *Ladder* do FB, que implementa a ação do SFC *Main Action_Sup,* é ilustrada na Figura 30. Segundo Vieira (2007), deve ocorrer, com essa ação, a atualização do estado ativo dos supervisores e respectivas ações de controle tratadas no ciclo de atualização anterior do CLP. Isto é feito na primeira linha da seção de código, quando o FB *MS* é chamado.

Outra imposição do método é a desativação condicional dos sinais de indicação de evolução de um modelo. Isto é visto na terceira linha de código, sendo que a negação de *CMD_A1*, referente ao evento controlável α_1, desativa a variável *G1_EV,* que sinaliza a ocorrência de algum evento em G_1. O comando pode ser enviado ao AD ou a um CLP Local correspondente. A desativação da variável que sinaliza a evolução de um subsistema só deve ser desativada quando o Procedimento Operacional Virtual (POV) ou Real (POR) correspondente receber o comando; portanto, a segunda linha de código representa a codificação do reconhecimento de recebimento do comando.

Observa-se a existência da variável *SIM_CMD_A1*. Essa variável é

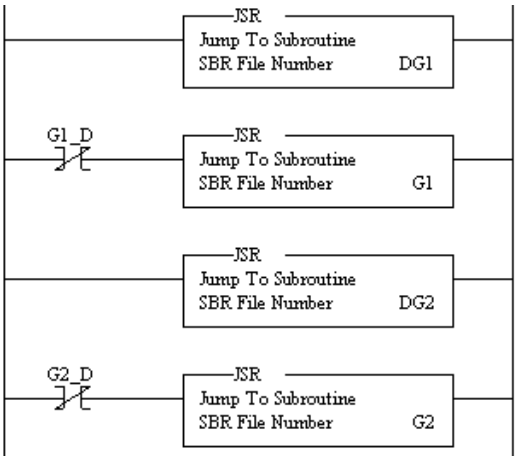

Fig. 29. Seção de código referente ao FB *PS*.

responsável por habilitar ou não o envio do *CMD_A1* para o CLP Local que contém o POR correspondente. No Ambiente Dinâmico (AD), há uma interface para se escolher quando o POR está conectado, então, uma variável no AD torna-se verdadeira. Essa variável é associada à variável do CLP Central (*SIM_CMD_A1*). Então, quando ela for verdadeira, o comando só poderá ser desativado quando o POR no CLP Local receber o comando e informar ao CLP Central através de *MSG_CMD_A1/DN* (comando enviado e reconhecido pelo POR correspondente no CLP Local). Se a variável é falsa, então o reconhecimento do recebimento de comandos é feito pela Plataforma de Simulação, que ativa uma variável associada a *ACK_CMD_A1* (comando enviado e reconhecido pelo POV correspondente na simulação). Isso permite a desativação do comando e, também, do sinal *G1_EVT*. A forma de implementação referente a G_2 é similar à forma de G_1.

Outro ponto importante do método de Vieira (2007) para a

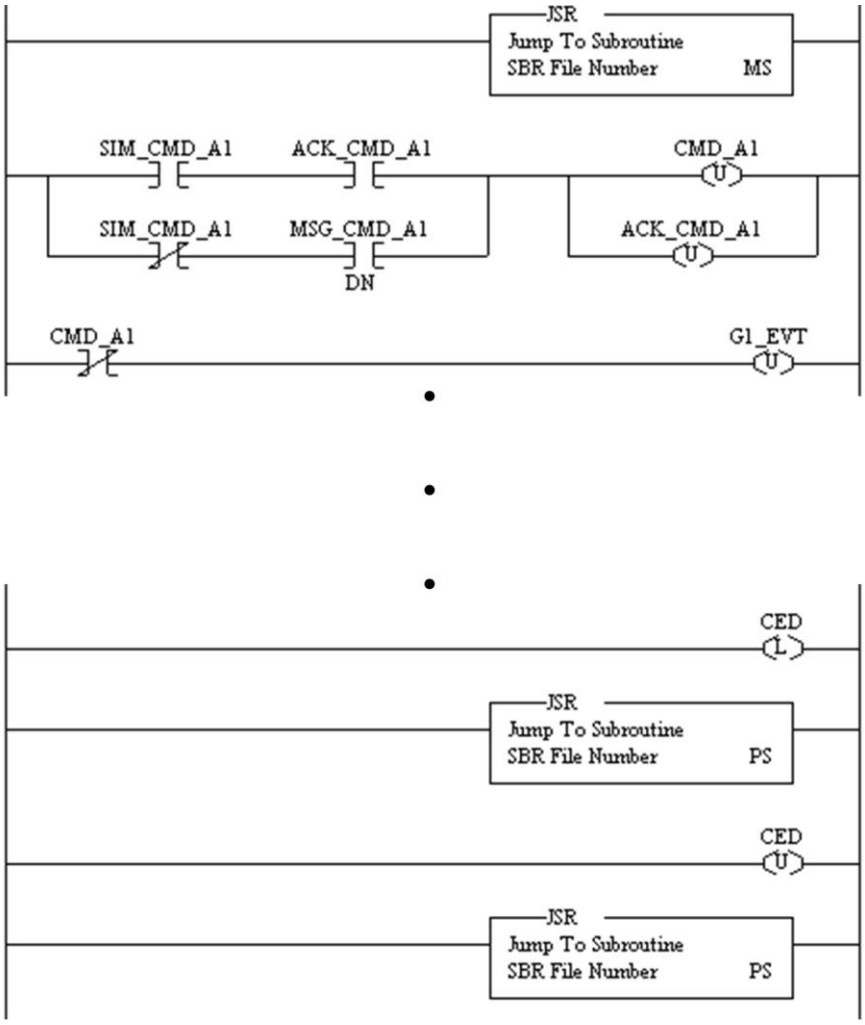

Fig. 30. Seção de código referente à ação do SFC Main *Action_Sup*.

Action_Sup é a suspensão do tratamento de eventos controláveis, para priorizar o tratamento de eventos não-controláveis. A primeira linha da segunda parte de código da Figura 30 ilustra a ativação da variável *CED* utilizada para tal priorização. Essa mesma variável já foi vista na seção de código da Figura 24. Observa-se que esta variável sempre é ativada com retenção quando o FB *Action_Sup* é chamado.

A linha inferior à ativação da variável *CED* da seção de código realiza a chamada do FB *PS*, outro ponto abordado por Vieira (2007). Quando ocorre um evento não-controlável e a resposta correspondente possui valor maior que zero, a ocorrência do evento é sinalizada. A próxima linha da programação desativa a variável *CED,* permitindo que o tratamento de eventos controláveis seja possível, quando se chama novamente o FB *PS*, observado na última linha de código.

4.2.7 Tratamento do envio de comandos no CLP Central

Durante o passo de simulação da etapa de Implementação do Ciclo de Desenvolvimento, o CLP Central somente envia comandos para a Plataforma de Simulação. De maneira a ilustrar isso, foi inserida a seção de código *Ladder* da Figura 31. Verifica-se que a variável *SIM_CMD_A1* novamente aparece, pois o envio

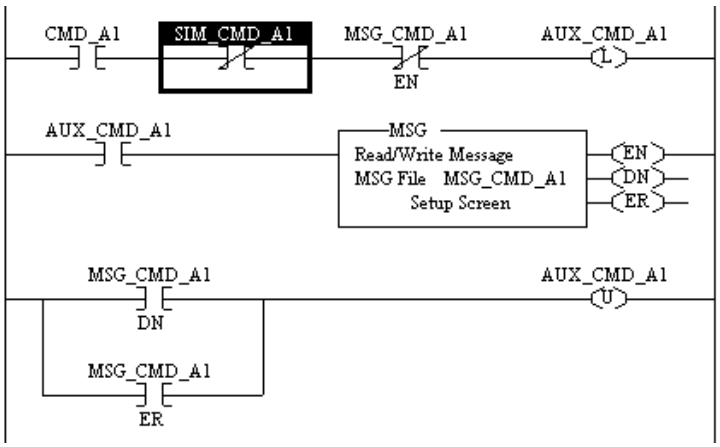

Fig. 31. Seção de código para tratamento de envio de comandos, especificamente do referido comando $Cmd\alpha_1$.

de um comando para um CLP Local correspondente só pode ocorrer quando um Procedimento Operacional Real (POR) implementado neste CLP Local está conectado. Então, quando a variável *SIM_CMD_A1* é verdadeira, o CLP Central fica impedido de enviar o comando ao CLP Local que contém o POR correspondente. Neste caso, o Ambiente Dinâmico (AD) lê o valor do comando diretamente na memória do CLP Central, através de *driver* de comunicação, e escreve na Plataforma de Simulação.

Considera-se que um POR_1 referente a G_1 foi implementado em um CLP Local, pois a Arquitetura de Controle Supervisório (ACS) e o Procedimento Operacional Virtual (POV) correspondentes foram validados através da simulação. Então, deseja-se enviar os comandos para o CLP Local. A Figura 31 ilustra exatamente como devem ser as linhas de programação para se executar tal tarefa. Se a variável *CMD_A1* estiver ativada, e as variáveis *SIM_CMD_A1* e *MSG_CMD_A1/EN* (bloco de mensagem ativado), não estiverem ativas, a variável *AUX_CMD_A1* é ativada com retenção. Essa variável ativa o bloco de mensagem que está configurado para escrever o comando no CLP Local correspondente. Quando o ciclo de envio de mensagem é completado, a variável *MSG_CMD_A1/DN* torna-se ativa e *AUX_CMD_A1* é desativada, permitindo que o comando seja enviado apenas uma vez. Caso *SIM_CMD_A1* esteja ativado, significa que o usuário deseja somente executar a simulação, portanto, o AD lê o valor da variável diretamente na área de memória do CLP Central.

4.3 Implementação do AD

Nas seções anteriores, a implementação da Arquitetura de Controle Supervisório (ACS) proposta por Queiroz *et al* (2001) foi tratada seguindo o método de implementação de código para CLP segundo Vieira (2007). Nesta seção, é tratada a maneira como foi desenvolvida a aplicação do Ambiente Dinâmico (AD), para o exemplo proposto de duas máquinas. Uma visão geral da interface com o usuário, para a escolha entre o modo de Simulação e quando um POR está conectado ao CLP Central (modo Ambiente Real), é ilustrada na Figura 32. Ainda é

possível se observarem campos que permitem ao usuário o acompanhamento dos valores que o AD recebe e envia para o CLP Central e para a Plataforma de Simulação. Os procedimentos para se chegar nesta interface gráfica com o usuário são descritos a seguir.

As figuras 14 e 15 mostram que o AD está inserido entre os níveis SP e PO da ACS, pois é nesse ponto que ocorrem a leitura e escrita de comandos e respostas entre os dois níveis. A forma desta comunicação, utilizada para o exemplo, está em maiores detalhes ilustrada na Figura 33.

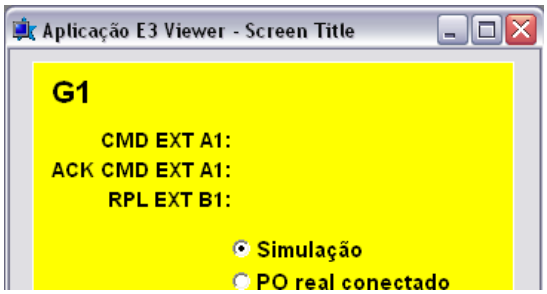

Fig. 32. Interface com o usuário para a aplicação do AD.

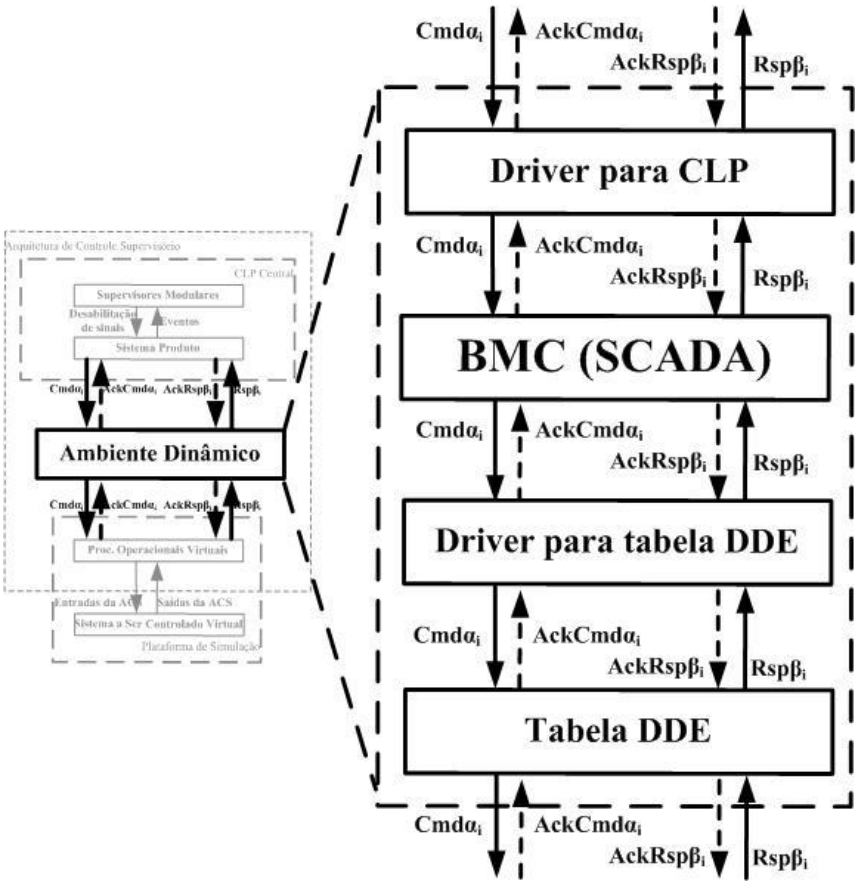

Fig. 33. Detalhamento para as comunicações entre CLP Central e Plataforma de Simulação, com $i = 1, 2$.

O AD é composto por uma plataforma SCADA, uma tabela DDE (*Dynamic Data Exchange*) e *drivers* de comunicação para comunicação do CLP Central com o SCADA e deste com a tabela DDE. Como *software* da plataforma SCADA, foi escolhido o Elipse E3, porque tem a característica de poder formar biblioteca de modelos e por possuir *drivers* de comunicação com periféricos e Base de Dados (BD). A tabela DDE foi escolhida devido ao fato de que o Elipse E3 tem *driver* de comunicação com este tipo de BD e por propiciar uma troca de dados mais rapidamente. Como Plataforma de Simulação, o *software* Arena, da Rockwell Software, foi selecionado, pois, além de permitir se validar a ACS através da dinâmica dos modelos virtuais, é possível se validarem os dados de produção, através da sua capacidade de geração de relatórios a respeito do sistema de manufatura a ser implementado fisicamente (Diogo *et al*, 2008).

4.3.1 Comunicação entre o CLP Central e o AD

Um *driver para* CLP, disponibilizado pela própria Elipse Software, é inserido na aplicação desenvolvida no E3 (a aplicação é o Ambiente Dinâmico) como suporte de comunicação entre o CLP Central e o AD. Com esse *driver,* é possível se associarem as variáveis do CLP com as variáveis do AD, implementadas no Elipse E3. Primeiramente, retomando à variável *SIM_CMD_A1,* que determina se o CLP Central escreve o comando respectivo no CLP Local ou na Plataforma de Simulação. Esta variável torna-se verdadeira no CLP Central quando o usuário escolhe, na aplicação do AD, a opção "*Simulação*", impedindo, assim, que o CLP Central escreva o comando correspondente no CLP local. Caso contrário, se o usuário escolheu a opção "PO real conectado", é permitida a comunicação entre os CLP.

O AD lê ciclicamente comandos do CLP Central, e, também ciclicamente, escreve este valor em um campo da tabela DDE. Então, quando um comando é ativado no CLP Central, por exemplo, *CMD_A1*, ele é escrito no campo reservado da tabela DDE. Portanto, a variável do CLP referente ao comando está associada a uma *tag* (*CMD_EXT_A1*), ou seja, uma variável no AD. Conforme

descrito anteriormente, deve haver o reconhecimento de recebimento do comando por parte da plataforma de simulação, para que o programa implementado, seguindo o método proposto por Vieira (2007), seja executado de forma correta. A aplicação do AD recebe este reconhecimento em outra *tag* interna (*ACK_CMD_EXT_A1*), que esta está associada à variável de reconhecimento de comando no CLP Central (*ACK_CMD_A1*).

Considera-se que as respostas são geradas pela Plataforma de Simulação. Como exemplo para este caso, considera-se a resposta de G_1, $Rsp\beta_1$. O AD possui uma *tag* denominada *RSP_EXT_B1,* que está associada à variável do CLP Central *RSP_B1*. De maneira análoga ao reconhecimento de recebimento de comandos, também deve haver o reconhecimento de recebimento de respostas, para se atualizar o valor das variáveis nos respectivos campos da tabela DDE. Porém, outra *tag* denominada *ACK_RSP_EXT_B1* está associada à mesma variável do CLP *RSP_B1*. Contudo, a *tag* lê o valor da variável do CLP e escreve no campo determinado da tabela DDE, com o intuito de atualizar seu valor e permitir que a Plataforma de Simulação escreva um novo valor de resposta no campo.

A codificação dessas associações é realizada de forma semelhante, para gerar o modelo no AD referente a G_2. Esses modelos podem ser armazenados na biblioteca do Elipse E3, permitindo, assim, que sua codificação seja aproveitável para outras aplicações, e, também, somente para armazenamento de modelos, quando não mais necessários. Por outro lado, o modelo pode ser reinserido em um futuro que exija o mesmo modelo para comunicação. Pode-se fazer, então, analogia desta biblioteca com a Biblioteca de Modelos de Comunicação (BMC), proposta por este trabalho.

4.3.2 Comunicação entre o AD e a Plataforma de Simulação

Um *driver* DDE serve de comunicação entre o Ambiente Dinâmico (AD) e a tabela DDE (*Dynamic Data Exchange*). Neste *driver*, *tags* são inseridas e associadas às *tags* do *driver* CLP. Isto é feito para que os valores trocados entre o CLP Central e a Plataforma de Simulação sejam escritos (atualizados) na tabela

DDE, ilustrada na Figura 34. A Plataforma de Simulação realiza diversas consultas e escreve dados nesta tabela.

A coluna denominada "*Real*" verifica quando um CLP Local, onde está implementado um Procedimento Operacional Real (POR), está conectado ao CLP Central através de Tecnologias de Controle e Comunicação (TCC). Caso o valor de uma célula nesta coluna seja verdadeiro, a Plataforma de Simulação fica impedida de gerar respostas e escrever no CLP Central através do AD. Caso contrário, o modelo de simulação respectivo tem permissão para gerar respostas.

A Plataforma de Simulação consulta as células da coluna "*cmd*" e analisa quando ocorre o novo comando. Quando o processo de simulação é iniciado, a coluna "*ackcmd*" tem todos os valores das variáveis colocados em nível lógico verdadeiro. Uma combinação lógica dos valores da coluna "*cmd*" e da coluna "*ackcmd*" dita se o comando é novo ou não:

i) Se *cmd = 0* e *ackcmd = 0*, então o Procedimento Operacional Virtual (POV) não é iniciado.

ii) Se *cmd = 0* e *ackcmd = 1*, então houve reconhecimento do recebimento por parte do CLP Central sobre algum comando enviado anteriormente, ou a Plataforma de Simulação está sendo executada pela primeira vez. O POV não é iniciado.

iii) Se *cmd = 1* e *ackcmd = 0*, então o CLP Central enviou um comando, mas a Plataforma de Simulação ainda não realizou o procedimento de envio do reconhecimento de recebimento. O POV é iniciado.

iv) Se *cmd = 1* e *ackcmd = 1*, então houve reconhecimento do recebimento de comando e o POV já foi iniciado. Enquanto esses valores se mantiverem, o POV não será mais iniciado, somente quando *cmd* voltar ao nível lógico falso e ocorrer um novo comando, tornando este valor verdadeiro.

	A	B	C	D	E
1					Are
2		Real	cmd	rpl	ackcmd
3	PO1	0	0	0	0
4	PO2	0	0	0	0
5					

Fig. 34. Tabela DDE: Base de Dados para que o AD troque dados entre o CLP Central e a Plataforma de Simulação.

A coluna “*rsp*” está reservada a receber valores da Plataforma de simulação e do AD. Quando uma resposta é gerada por um modelo de simulação, seu valor é escrito no respectivo campo da coluna “*rsp*”. Este valor é lido pela respectiva *tag* associada no AD e transmitido à memória do CLP Central, informando, assim, a ocorrência de um evento não-controlável. O reconhecimento de recebimento é realizado no CLP Central, atualizando o valor da variável para nível lógico falso, e retransmitido por uma *tag* *AckRspB1* ao mesmo campo da tabela DDE, permitindo, assim, que novas respostas sejam escritas nas células da coluna “*rsp*”.

4.4 A Plataforma de Simulação

A plataforma de simulação foi desenvolvida no *software* Arena. Uma visão geral é ilustrada na Figura 35, na qual o bloco “*Create*” gera entidades a cada 200ms, funcionando como um *clock* da simulação. O bloco “*VBA*” contém a codificação em *Visual Basic for Applications,* para determinar quando novos comandos são gerados pelo CLP Central e encaminhá-los aos respectivos Procedimentos Operacionais Virtuais (POV), conforme estrutura da Figura 18. As entidades criadas, que não se refiram a nenhum comando novo, são desviadas ao bloco “*Idle*” para serem descartadas. Já as entidades que se referem a comandos novos são desviadas ao bloco “*PO1*” ou “*PO2*”, a fim de iniciar o respectivo POV. Os blocos “*Process 1*” e “*Process 2*” são submodelos na Plataforma de Simulação. Um exemplo de submodelo é ilustrado na Figura 36 e explicado a seguir.

Um submodelo é composto pelo conjunto de passos de um POV. Na Figura 36, é observado um primeiro bloco “*VBA*” que verifica se o AD informa quando um subsistema físico real está conectado; caso positivo, o modelo é impedido de gerar respostas, caso contrário, é permitido. O bloco “*Process 11*” é o POV propriamente dito. O segundo bloco “*VBA*” contém o código necessário para gerar a resposta respectiva. Esta resposta é gravada no campo correspondente da tabela DDE.

4.5 Alteração do exemplo de manufatura proposto

Para procedimento de verificação da flexibilidade do Ambiente Dinâmico (AD), uma modificação no sistema de manufatura da Figura 20 é sugerida. Considera-se que uma nova máquina M_3 foi inserida no sistema de manufatura da Figura 20, entre as máquinas M_1 e M_2. Há, agora, dois *buffers* B_1 e B_2. A máquina M_3 realiza a transferência de peças de B_1 para B_2, conforme Figura 37. Verifica-se a necessidade de se iniciar o Ciclo de Desenvolvimento, para modelagem da terceira máquina. Pelos procedimentos formais, obteve-se o modelo G_3, no qual: o estado 0, estando ativo, significa que o manipulador robótico está em repouso; no estado 1,

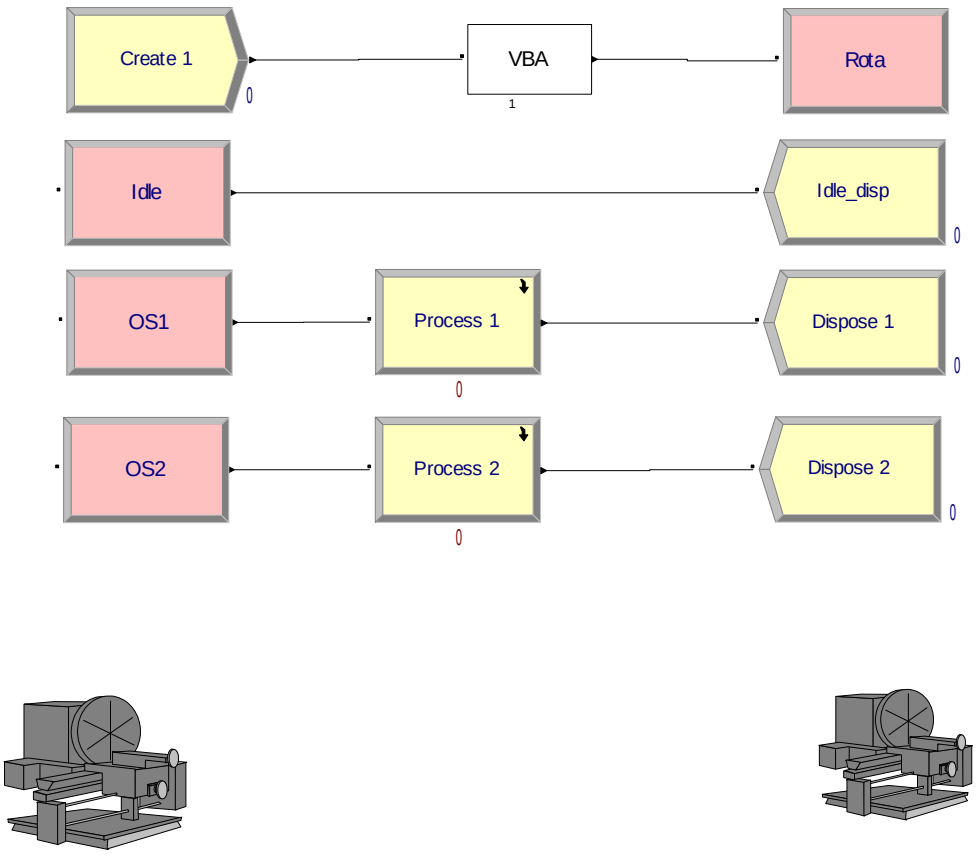

Fig. 35. Plataforma de Simulação.

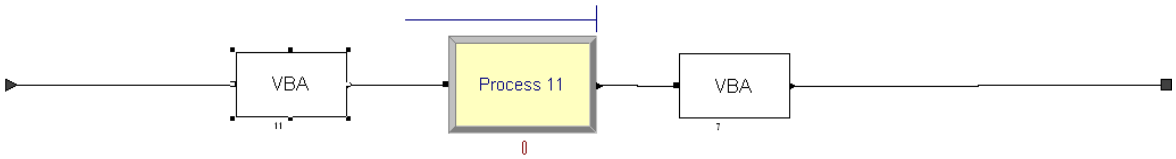

Fig. 36. Submodelo da Plataforma de Simulação.

levando peça de B_1 para B_2, e, no estado 2, retornando à posição de repouso.

Ainda a modelagem das novas especificações para os dois *buffers* foi necessária. De forma semelhante à especificação do sistema de manufatura original, os *buffers* não podem ter mais de uma peça, ou o robô não pode tentar retirar peça inexistente dos mesmos. Dessa forma, foram obtidas as especificações comportamentais da Figura 38. Tais especificações resultaram nos Supervisores Modulares (SM) reduzidos, representados na mesma figura.

Diante desse novo panorama, mudanças foram necessárias na programação do CLP Central, com a inserção de um novo FB referente ao modelo G_3, bem como variáveis internas no CLP Central e a inserção do FB da suspensão do tratamento de eventos do referido modelo. Foi realizada a exclusão do FB do SM original, para inserção dos novos SM referentes às novas especificações. Verifica-se, desta forma, que a programação modular do CLP Central permite o reaproveitamento de código, tornando o projeto da Arquitetura de Controle Supervisório (ACS) mais rápida.

O novo panorama também dita mudanças na aplicação do AD.

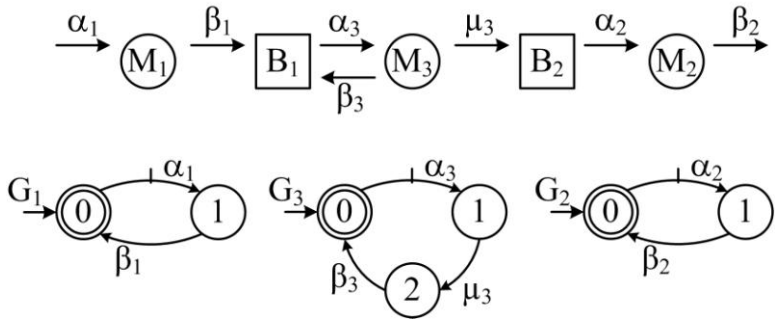

Fig. 37. Modelos de subsistemas para o exemplo modificado.

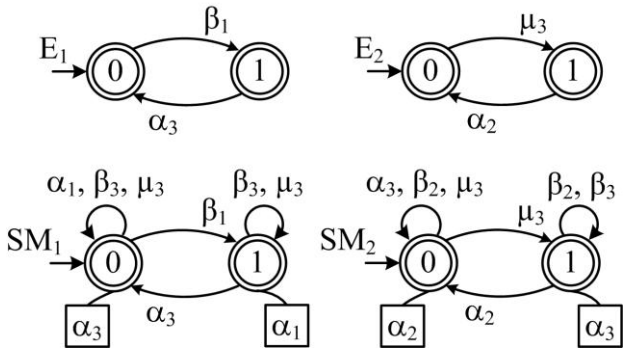

Fig. 38. Especificações e supervisores modulares reduzidos para o sistema de manufatura modificado.

Porém, verifica-se que os modelos de comunicação para G_1 e G_2 permanecem na mesma forma, sendo necessária somente a inserção do modelo de comunicação referente ao modelo G_3, no qual, novas *tags* referentes a comandos e respostas serão associadas às novas variáveis do CLP Central e aos campos da tabela DDE. Esta, por sua vez, necessita de mais campos para efetuar a leitura e escrita de comandos e respostas entre AD e Plataforma de Simulação, mas os campos originais não precisam ser alterados.

Na Plataforma de Simulação, é inserido um novo modelo para simulação da nova máquina e o seu respectivo submodelo. As linhas de código para leitura do comando referente à nova máquina são acrescentadas ao bloco "*VBA*", assim como as linhas de código para determinação das duas respostas referentes aos eventos não-controláveis. Porém, os modelos originais não são modificados.

4.6 Conclusão do capítulo

Este capítulo mostrou o desenvolvimento de uma aplicação hipotética de um sistema de manufatura, para validação da proposta da dissertação. Posteriormente, este sistema foi alterado para provar a economia de tempo em alterações do projeto inicial.

Através da inserção da terceira máquina, provou-se que a flexibilidade do AD e demais TCC estão presentes. Isto prova que o AD pode ser uma ferramenta poderosa para projeto de sistemas de manufatura flexíveis utilizando a TCS e ACS como procedimentos formais de suporte. Os ganhos, neste caso, podem ser notados pela economia de tempo nos projetos. Com isto, pode-se obter vantagem quando novas necessidades de produção são desejadas, conseguindo atender aos requisitos dos clientes em menor tempo.

5 CONCLUSÕES

Esta dissertação demonstrou a preocupação da academia e de diversos setores industriais com relação à importância da agilidade no desenvolvimento de projetos para Sistemas Flexíveis de Manufatura (FMS). Um atraso em relação aos concorrentes resulta em perdas incomensuráveis. Percebeu-se a existência de diversas ferramentas teóricas e computacionais como suporte de projeto, porém, raras são as integrações dessas ferramentas para validar a Arquitetura de Controle Supervisório (ACS) de Queiroz *et al* (2001), aplicada a controladores industriais que agem sobre o sistema de manufatura.

É perceptível que todas as metodologias aplicadas para o desenvolvimento do Ambiente Dinâmico (AD), como a Teoria de Controle Supervisório (TCS), de Ramadge e Wonham (1989), a ACS, de Queiroz *et al* (2001), o conceito de *Hardware-in-the-Loop*, o método de implementação de ACS em CLP, de Vieira (2007), o Ciclo de Desenvolvimento, de Busetti e Santos (2006), e as demais aplicadas a esta dissertação, ajudaram a transformá-lo em uma poderosa ferramenta para integrar Controladores Lógico Programáveis (CLP) à plataformas de simulação. Essa integração objetivou à validação da ACS implementada em CLP através de simulação. Esta, por sua vez, permite que os FMS sejam testados computacionalmente para procurar erros que inviabilizem sua imediata implementação física. Isto permite que o processo de projetar tais sistemas seja ágil, a construção de protótipos diminuída ou até mesmo eliminada, o que reduz os custos de projeto.

Ainda foi possível observar que a implementação de parte da ACS em CLP eliminou a necessidade de se implementar uma metodologia de tradução da ACS para uma plataforma de simulação. A simulação da ACS, e a implementação progressiva dos dispositivos de controle, são caracterizadas como ferramentas importantes para consolidação da metodologia. Estas ferramentas permitem a redução do tempo no projeto de sistemas de manufatura automatizados e integrados. Esta característica aumenta, sem dúvida, a confiabilidade do projeto global.

O estudo de um sistema de manufatura hipotético permitiu que sua ACS fosse testada computacionalmente e validada. Percebeu-se que a integração do CLP Central à Plataforma de Simulação foi possível devido à inserção de Tecnologias de Controle e Comunicação (TCC) existentes. Os métodos para desenvolvimento do AD permitiram que os eventos, gerados pelo CLP Central, fossem escritos na Plataforma de Simulação, e que esta respondesse com outros eventos, que foram escritos no CLP Central. A ACS implementada, então, pôde ser verificada através da simulação estimulada por eventos reais, e, posteriormente, validada. A partir disso, os subsistemas físicos reais do sistema de manufatura foram gradualmente inseridos, até atingirem sua completude.

Na Plataforma de Simulação, além de se validar a ACS através da dinâmica virtual dos modelos testados, é possível se verificar se os requisitos de produção atingiram as exigências submetidas pelo mercado. Neste caso, o setor de tomada de decisões de uma empresa pode agir, pois a Plataforma de Simulação permite que dados reais referentes aos subsistemas de manufatura sejam inseridos, permitindo que os requisitos de demanda sejam avaliados.

Outro ponto importante a ser notado é que foi possível manter a modularidade dos FMS e a distribuição espacial de seus controladores. Pôde ser visto que a implementação em CLP, a implementação no AD e na Plataforma de Simulação também foram realizadas de forma modular. Isto permitiu que modelos menores fossem desenvolvidos para cada aplicação. Tais modelos podem ser armazenados na Biblioteca de Modelos de Comunicação (BMC). Esses modelos podem ser reutilizados quando uma mudança for necessária no sistema de produção, o que permite se visualizar a flexibilidade atingida pela metodologia proposta por este trabalho.

Apesar de vários pontos positivos serem encontrados, pontos negativos também surgiram, devido a limitações verificadas no desenvolvimento da aplicação do AD. A existência de tais limitações permite presumir a continuidade do trabalho que foi desenvolvido. Uma desvantagem da aplicação da metodologia está associada à implementação progressiva da ACS. Isto porque a técnica faz uso de TCC, e vários problemas relacionados à integração e automação ocorrem durante a implementação. A principal causa é a adoção de sistemas proprietários pelos fabricantes no mercado, como protocolos de comunicação, redes industriais, *softwares* e outros. A desvantagem encontra-se na integração de sistemas

heterogêneos, nos quais, o estabelecimento progressivo da comunicação com vários dispositivos diferentes é necessário.

Outra limitação enquadra-se no uso de plataformas computacionais que são executadas em sistemas operacionais que não operam em tempo real, e a utilização de protocolos de rede não-determinísticos. Portanto, a leitura e escrita de eventos praticada pelo AD apresentam atraso devido ao processamento e tráfego das informações na rede utilizada. Isto resultará em um mascaramento dos reais valores de tempo obtidos com a Plataforma de Simulação. Isto justifica a continuidade do trabalho, inicialmente tratado por Diogo *et al* (2008), com o objetivo de diminuir o não-determinismo encontrado nas redes industriais e a deficiência em trabalhar em tempo real de certos sistemas operacionais.

Ainda como continuidade do trabalho, melhorias nos canais de comunicação do AD podem ser propostas, pois foi verificada concorrência de comunicação quando a Plataforma de Simulação e um CLP Local solicitam uma consulta na memória do CLP Central. A perda da ocorrência de eventos por uma das partes foi verificada. Quando o sistema está no passo de Simulação do Ciclo de Desenvolvimento, este problema não é detectado, pois só há comunicação do AD e CLP Central.

Outra proposta seria estender e aplicar a metodologia a sistemas operacionais de tempo real, com o intuito de realizar a simulação em tempo real de Sistemas Flexíveis de Manufatura.

Em linhas gerais, verificou-se que o desenvolvimento de aplicações no Ambiente Dinâmico permite que a validação de Arquiteturas de Controle Supervisório seja realizada com agilidade. O Ambiente Dinâmico também é notável, pela sua flexibilidade em reutilizar os modelos da Biblioteca de Modelos de Comunicação.

6 REFERENCIAS BIBLIOGRÁFICAS

ADBI, Mohammad Reza; LABIB, Ashraf W. **A design strategy for reconfigurable manufacturing systems (RMSs) using analytical hierarchical process (AHP): a case study**. International Journal in Production Research. Taylor & Francis Group. Vol. 41, nº 10, p. 2273-2299, Manchester, UK, 2003.

BABIC, B. **Axiomatic design of flexible manufacturing systems**. International Journal in Production Research. Taylor & Francis Group. Vol. 37, nº 5, p. 1159-1173, Belgrade, Yugoslavia, 1999.

BERTO, Rosa Maria Villares de Souza; NAKANO, Davi Noboro. **Metodologia da Pesquisa e a Engenharia de Produção**. ENEGEP 1998.

BONFE, Marcello; DONATI, Cláudio; FANTUZZI, Cesare. **An Application of Software Design Methods to Manufacturing Systems Supervision and Control**. Proceedings of the 2002 IEEE International in Conference on Control Applications. Glasgow, Scotland, UK.

BOYD, John; THEYYUNNI, Roger. **Development of A Real-Time Simulation System**. Embedded Systems Programming. ADI. 2006.

BRACKENHAMMER, Eckart; SCHMIDT, Eberhard. **Virtual instrumentation drives hardware-in-the-loop simulation**. Global Report: Industrial Automation and Control. 2005.

BRUNS, F. Wilhelm; ERBE, Heinz-H. **Mixed reality with hyper-bonds – A means for remote labs**. Control Engineering Practice. Elsevier. Berlim, Germany, 2006.

BRYMAN, Alan. **Research methods and organization studies**. P. 283. Unwin Hyman, London, 1989.

BULLOCK, Darcy; JOHNSON, Brian; WELLS, Richard B.; KYTE, Michael; LI, Zhen. **Hardware-in-the-loop simulation**. Transportation Research Part C. Elsevier. Pág. 73-89, West Lafayette, 2004.

BUSETTI, Marco Antonio. **Prozeβkopplung mittels einer VMEbus-basierten Hardware-Plattaform für eine verteilte Simulationsumgebung mechatronischer Systeme**. Tese de Doutorado. Universidade de Paderborn. Alemanha, 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 2006. Curitiba, Brazil.

CAI, Xiujun; VYATKIN, Valeriy; HANISCH, Hans-Michael. **Design and Implementation of a Prototype Control System According to IEC 61499**. IEEE. 2003.

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

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

CURY, José Eduardo Ribeiro. **Teoria de Controle Supervisório de Sistemas a Eventos Discretos**. Quinto Simpósio Brasileiro de Automação Inteligente. Canela, Brazil, 2001.

1.

DEPPE, M.; ZANELLA, M.; ROBRECHT, M.; HARDT, W. **Rapid prototyping of real-time control laws for complex mechatronic systems: a case study**. The Journal of Systems and Software. Elsevier. P. 263-274, Germany, 2004.

Dessault Systemes. Disponível em www.3ds.com (ultima visualização em maio de 2007).

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, Korea. 2008.

Discovery Channel. Disponível em www.discoverybrasil.com (ultima visualização em maio de 2007).

Elipse Software. Disponível em www.elipse.com.br (ultima visualização em maio de 2007).

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.

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.

GOUVÊA DA COSTA, Sergio Eduardo. **Definições Metodológicas e Métodos de Pesquisa**. PPGEPS/PUCPR. 2006.

HELLGREN, Anders; LENNARTSON, Bengt; FABIAN, Martin. **Modelling and PLC-based Implementation of Modular Control Supervisory**. Proceedings of the Sixth International Workshop on Discrete Event Systems (WODES'02). IEEE Computer Society. Götenborg, Sweden, 2002.

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.

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, 2006.

HOLST, L.; RANDELL, L.; BOLMSJÖ, G. **Integrated Development of Manufacturing Systems Using Simulation – Proposing the Fundamentals for a Joint Research Project.** In: Proceedings of the 33rd CIRP International Seminar on Manufacturing Systems: The Manufacturing Systems in its Human Context – A Tool to Extend the Global Welfare, 5-7, Stockholm, Sweden. 2000.

HOLST, L., BOLMSJÖ, G.; RANDELL, L.; NORGREN, J. **Integrating Simulation into Manufacturing System Development: A Methodological Framework**. In: Proceedings of the Twelth Annual Conference of the Production and Operations Management Society, POM-2001, Orlando, USA. 2001

IEC 61499 Internation Standard. 2005.

IEC 61131-3 Programmable Controllers languages. 2003.

IEC. **International Standard IEC 61131-5, Programmable Controllers – Part 5: Communications**. 2000.

KOCIJAN, Jus; KARBA, Rihard. **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. **Modeling and Control of Logical Discrete Event Systems**, Kluwer Academic Publishers, 1995.

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

LAUZON, S.C.; MA, A. K. L.; MILLS, J. K.; BENHABIB, B. **Application of Discrete-Event-System Theory to Flexible Manufacturing**. 1995 IEEE International Conference on Robotics and Automation. Nagoya, Japan.

LEE, K. C.; LEE, S. **Performance evaluation of switched Ethernet for real-time industrial communications**. Computer Standards & Interfaces. Elsevier, n. 24, pp. 411-423, 2002.

LI, 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. PUCPR, 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. Vol. 44, nº 1, p. 65-81, South Africa, 2006.

NAKANO, Dani Noboru; FLEURY, Afonso Carlos Corrêa. **Métodos de Pesquisa na Engenharia de Produção**. ENEGEP 1996.

National Instruments. Disponível em www.ni.com/labview (ultima visualização em maio de 2007).

PLUMMER, A. R. **Model-in-the-loop testing**. Proceedings of IMechE Vol. 220 Part I: J. Systems and Control Engineering, 2006.

QUEIROZ, Max H. de; CURY, José E. R. **Modular Supervisory Control of Large Scale Discrete-Event Systems**. Proceedings of the 5th International Workshop on Discrete Event Systems: Analysis and Control. Kluwer Academic Publishers, Ghent, Belgium, p. 103-110, 2000a.

QUEIROZ, Max H. de; CURY, José E. R. **Modular Control of Composed Systems**. Proceedings of Control Conference. Chicago, USA, 2000b.

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

QUEIROZ, Max H. de; CURY, José E. R. **Synthesis and Implementation of Local Modular Supervisory Control for a Manufacturing Cell**. Proceedings of the Sixth International Workshop on Discrete Event Systems. IEEE Computer Society, 2002.

RAMADGE, Peter J. G.; WONHAM, W. Murray. **The Control of Discret Event Systems**. Proceedings of the IEEE, vol. 77, nº 1, January, 1989.

Rockwell Software. Disponível em www.rockwellautomation.com/rockwellsoftware (ultima visualização em maio de 2007).

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

The Boeing Company. Disponível em www.boeing.com (última visualização em maio de 2007).

The MathWorks. Disponível em www.mathworks.com (última visualização em maio de 2007).

ZAMBALDI, M.; ECKER, W. **How to Bridge the Gap between Simulation and Test**. ITC International Test Conference. Germany, 2004.

UGS. Disponível em www.ugs.com (ultima visualização em maio de 2007).

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. **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.

WADHWA, S.; RAO, K. S.; CHAN, F. T. S. **Flexibility-enabled lead-time reduction in flexible systems**. International Journal of Production Research, vol. 43, nº 15, p.3131-3162. 2005

WESTBROOK, Roy. **Action Research: a new paradigm for research in production and operations management**. International Journal of Production and Operations Management, vol 15, n' 12, p.6-20. 1995.

WONHAM, W. M.; RAMADGE, P. J. **On the supremal controllable sublanguage of a given language**. SIAM Journal of Control and Optimization, v. 25, n. 3, p. 637-659, 1988.

Livros Grátis

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

Milhares de Livros para Download:

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