ROBSON MARINHO DA SILVA

CONTROLE DE SISTEMAS RECONFIGURÁVEIS DE MANUFATURA

São Paulo
2016

ROBSON MARINHO DA SILVA

CONTROLE DE SISTEMAS RECONFIGURÁVEIS DE MANUFATURA

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de doutor em Ciências

São Paulo
2016

ROBSON MARINHO DA SILVA

CONTROLE DE SISTEMAS RECONFIGURÁVEIS DE MANUFATURA

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de doutor em Ciências

Área de concentração:
Engenharia de Controle e Automação Mecânica

Orientador:
Prof. Dr. Paulo Eigi Miyagi

Co-orientador:
Prof. Dr. Diolino José dos Santos Filho

São Paulo
2016

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

São Paulo, de de 2016

Assinatura do autor

Assinatura do orientador

Catálogo-na-publicação

da Silva, Robson Marinho

Controle de sistemas reconfiguráveis de manufatura / R.M. da Silva --versão corr.--São Paulo, 2016.
207 p.

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

1. Sistemas de manufaturas 2. Sistemas de controle 3. Arquitetura orientada a serviços 4. Sistemas multiagentes 5. Redes de Petri I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II. t.

DEDICATÓRIA

*à minha esposa Jomailsa
e aos nossos filhos Rafael e Felipe.*

AGRADECIMENTOS

A Deus por tudo que me tem concedido.

Ao professor Dr. Paulo Eigi Miyagi pela orientação, auxílio, incentivo constante e incondicional atenção.

Ao professor Dr. Diolino José dos Santos Filho pela co-orientação, inúmeras e valiosas contribuições, pelo direcionamento e acompanhamento deste trabalho.

Aos professores Dr. Fabrício Junqueira, Dr. Newton Maruyama, Dr. José Reinaldo Silva e Cristina Borba pelo apoio técnico e acadêmico.

A Maurício Blos, Edson Watanabe, Osvaldo Asato, Marcosiris, Caio Fattori, Jeferson Souza, Reinaldo Squillante, Célia Kano e a todos do LSA pelas sugestões, contribuições e amizade ao longo destes anos.

Ao professor Dr. José Barbosa (Instituto Politécnico de Bragança, Portugal) pela colaboração no desenvolvimento do exemplo de aplicação.

A Regianne e Marisa da secretaria do PPGEM pelo apoio administrativo/acadêmico.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) que auxiliaram financeiramente diferentes partes do trabalho.

Aos professores da Área de Engenharia Mecânica da UESC que me apoiaram, assumindo minhas atividades acadêmicas em parte do tempo que dediquei a este trabalho.

A minha esposa Jomailsa e aos nossos filhos Rafael e Felipe pelo apoio, incentivo e compreensão nos momentos de ausência.

Aos meus sogros João e Maria pela união e dedicação em todas as situações.

Aos meus pais Ribeiro e Olimpia responsáveis pela minha existência e educação.

Aos meus irmãos, amigos, parentes e a todos que colaboraram direta ou indiretamente na execução deste trabalho.

*“Ensinar não é transferir conhecimento
mas criar as possibilidades para
a sua própria produção ou a sua construção.”*

Paulo Freire

RESUMO

A capacidade de reconfiguração de sistemas de manufatura tem sido procurada pelas empresas para assegurar características de agilidade, eficiência e flexibilidade para atender as mudanças de tipo/quantidade de produtos, processos, recursos e, além disso, para assegurar a devida reação à ocorrência de falhas. Por outro lado, a “Indústria 4.0” impõe novos desafios para os sistemas de controle, tais como a integração de tecnologias de interação entre homem e máquina em cadeias de valor compondo uma rede de plantas industriais geograficamente dispersas. O controle de sistemas reconfiguráveis de manufatura deve considerar: (i) funcionalidades de sistemas distribuídos e dispersos, tais como agilidade de resposta às mudanças, autonomia e colaboração entre os componentes para alcançar os objetivos do sistema de forma conjunta; (ii) interfaces para sua interoperabilidade e portabilidade; (iii) modularização para facilitar a manutenção, expansão e atualização do sistema, evitando a sobreposição de escopos; e (iv) mecanismos de controle para supervisionar as ações e interações dos componentes, o diagnóstico e a tomada de decisão. O pleno atendimento a estes requisitos não é trivial e formalismos para o desenvolvimento de soluções devem ser adotados. Uma solução é combinar técnicas voltadas para sistema multiagente e *holon* com arquitetura orientada a serviço através de uma adequada técnica de modelagem usando extensões de rede de Petri: *Production Flow Schema* e *Input Output Place Transition*. Portanto, este trabalho propõe uma arquitetura de controle e o método de modelagem de seus componentes para sistemas reconfiguráveis de manufatura combinando estas técnicas e considerando os aspectos de personalização, convertibilidade, escalabilidade, modularidade, integrabilidade, diagnosticabilidade, interoperabilidade e colaboração entre os componentes do sistema de controle, inclusive do homem. Um exemplo de aplicação é apresentado para demonstrar a viabilidade da proposta e comprovar os resultados alcançados.

Palavras-chave: Sistema de manufatura. Reconfiguração de sistema. Arquitetura orientada a serviços. Sistema multiagente. Holon. Rede de Petri.

ABSTRACT

The reconfiguration ability of the manufacturing systems has been approached by companies to ensure agility, efficiency and flexibility characteristics to address the changes of type/quantity of products, processes and resources and, furthermore, to ensure proper reaction to the fault occurrence. On the other hand, the "Industry 4.0" imposes new challenges for control systems, such as interaction between man and machine into value chains composing a network of geographically dispersed industrial plants. The control of reconfigurable manufacturing systems should consider: (i) functionalities requirements of distributed and disperse systems, such as responsiveness to changes, autonomy and collaboration among components to achieve the global system aim; (ii) interfaces for interoperability and portability; (iii) modularity to facilitate maintenance, expansion and upgrade of the system, avoiding the overlapping of scopes; and (iv) control mechanisms to supervise the actions and interactions among components, diagnosis and decision making. The compliance with these requirements is not trivial and formalisms to develop solutions must be adopted. A solution is combining techniques based on holonic and multi-agent system with service-oriented architecture through appropriate modeling using Petri net extensions: Production Flow Schema and Input Output Place Transition. Therefore, this paper proposes control architecture and a method to model components for reconfigurable manufacturing systems, combining these techniques and considering aspects of customization, convertibility, scalability, modularity, integrability, diagnosability, interoperability and collaboration among control system components, including humans. An application example is presented to demonstrate the feasibility of the proposal and verify the results.

Keywords: Manufacturing system. System reconfiguration. Service-oriented architecture. Multi-agent system. Holon. Petri net

LISTA DE FIGURAS

Figura 1	Exemplo de mudanças de demanda e tipo de produto para sistemas de manufatura.	31
Figura 2	Exemplo de sistema produtivo disperso	33
Figura 3	Arquitetura de controle de sistemas de manufatura.	36
Figura 4	Relação entre falha-erro-defeito.	37
Figura 5	Diagrama conceitual de sistema de controle com reconfiguração. . . .	39
Figura 6	Elementos do PFS.	42
Figura 7	Evolução dinâmica derivada do disparo de uma transição. . . .	43
Figura 8	Elementos de uma RdP L/T.	43
Figura 9	Tela de trabalho do <i>web tool</i> IOPT.	47
Figura 10	Representação de holons e holarquias	49
Figura 11	Recursividade em sistemas holônicos.	49
Figura 12	Arquitetura básica de um agente.	50
Figura 13	Representação dos níveis de controle em um HMAS.	52
Figura 14	Abas do <i>software</i> NetLogo do exemplo jantar dos filósofos.	55
Figura 15	A plataforma JADE+WADE usada para implementar sistemas multia- gentes usando <i>workflow</i>	57
Figura 16	Interface para <i>web Services</i> de WADE.	58
Figura 17	<i>Workflow</i> do desenvolvimento da arquitetura de controle proposta. . .	69
Figura 18	Arquitetura e holarquias representando os níveis de controle em PFS. .	71
Figura 19	Composição de holarquias em PFS por meio de distribuidores representando interfaces.	74
Figura 20	Fases do tratamento de falha.	75
Figura 21	Estimação do tempo de reconfiguração	80
Figura 22	Evolução do fator autonomia.	81
Figura 23	Propagação de mensagem de ocorrência de uma falha.	82
Figura 24	PFS do método de especificação e implementação do controle de SRM. .	85
Figura 25	PFS da etapa modelagem conceitual.	87
Figura 26	PFS do modelo padrão para definição da ontologia.	88
Figura 27	Modelo conceitual padrão do holon de produto.	89
Figura 28	Exemplo de modelagem dos processos envolvendo um plano de produção representado por PrHs.	89
Figura 29	Modelo conceitual padrão do holon de tarefas.	90

Figura 30	Exemplo de modelagem de serviço de uma estação de trabalho (PWS).	91
Figura 31	Modelagem conceitual padrão do holon supervisor.	92
Figura 32	Modelo conceitual padrão do holon operacional.	92
Figura 33	Exemplo do PFS do objeto de controle representado pelo holon operacional.	93
Figura 34	PFS que representa a orquestração para execução de uma ordem de trabalho.	94
Figura 35	PFS que representa a orquestração para interação em caso de reconfiguração.	95
Figura 36	PFS da etapa modelagem da funcional.	96
Figura 37	IOPT do modelo padrão das funcionalidades do holon de produto. . .	98
Figura 38	Exemplo de IOPT da representação de produtos pelos PrHs.	99
Figura 39	IOPT do modelo padrão das funcionalidades do holon de tarefas. . .	100
Figura 40	Exemplo de modelo IOPT da funcionalidade do holon de tarefas de acordo com uma estratégia de controle.	101
Figura 41	IOPT do modelo padrão das funcionalidades do holon supervisor. . .	102
Figura 42	IOPT do modelo padrão das funcionalidades do holon operacional. . .	103
Figura 43	Esquema de modelo IOPT de um recurso industrial representado por holons	104
Figura 44	Exemplo de PFS da orquestração entre holons feito através de <i>asynchronous channels</i>	105
Figura 45	Exemplo de PFS/IOPT que ilustra o desenvolvimento da modelagem conceitual e dinâmica.	106
Figura 46	Exemplo de IOPT da modelagem dos mecanismos diagnosticador e decisor.	108
Figura 47	Exemplo de modelagem de um sistema de manufatura usando NetLogo.	109
Figura 48	PFS da etapa implementação.	110
Figura 49	Sistema modular de produção e diagrama da tela de trabalho do <i>software</i> NetLogo.	113
Figura 50	Desenho da DWS	117
Figura 51	Holons de produtos para o exemplo de aplicação em um MPS.	118
Figura 52	Refinamento da atividade [processos de negócio] do holon de estratégia.	118
Figura 53	Refinamento da estratégia de uma ordem de cliente e a seleção de fornecedores.	119

Figura 54	Orquestração entre PrHs, OpHs e THs para execução de um produto. .	120
Figura 55	PFS do serviço disponibilizado pela estação de distribuição.	121
Figura 56	Orquestração entre modelos conceituais dos processos interativos. . . .	121
Figura 57	Detalhamento e orquestração entre modelos de THs ilustrados na Figura 52.	122
Figura 58	Modelo da estratégia de controle da DWS.	123
Figura 59	Modelo composto de um conjunto de objetos de controles da MPS. . .	124
Figura 60	Exemplo da interação em IOPT entre TH e OpH através de <i>asynchronous</i> <i>channel</i>	126
Figura 61	Modelo diagnosticador para DWS.	127
Figura 62	Modelo de sub-diagnosticador para DWS.	128
Figura 63	Cenário de produção representado no Netlogo.	130
Figura 64	Exemplo de modelo em IOPT usado para descrição do mapeamento com o Netlogo.	135
Figura 65	Tela do Netlogo usada para configurar o gráfico do tempo de ocupação.	140
Figura 66	Simulação de cenários de produção no Netlogo.	141
Figura 67	Tela do JADE+WADE da atividade [seleção de fornecedor].	142
Figura 68	SFC para controle da DWS	143

LISTA DE TABELAS

Tabela 1	Comparativo entre sistemas de manufatura: reconfigurável, dedicado e flexível	32
Tabela 2	Tipos de arquitetura de controle	34
Tabela 3	Esquema de cores do <i>software</i> de edição/simulação de IOPT.	48
Tabela 4	Comparativo entre <i>software</i> para simulação de sistemas multiagentes. .	53
Tabela 5	Comparativo entre arquiteturas de controle.	75
Tabela 6	Mecanismo de negociação baseado no CNP (SMITH, 1980).	78
Tabela 7	Aspectos relacionados a requisitos de tolerância a falhas	86
Tabela 8	Lista de dispositivos de controle da DWS.	117

LISTA DE LISTAGENS

Listagem 1	Criação do agente <i>patch</i> que representa a DWS no Netlogo.	129
Listagem 2	Exemplo para definição do plano de produção.	131
Listagem 3	Supervisão em tempo de execução durante movimentação de PrHs na TrWS.	132
Listagem 4	Negociação entre holons para mudança do plano de produção e atendimento de novas demandas.	133
Listagem 5	Negociação entre holons para atendimento de novas demandas.	134
Listagem 6	Código para implementação do controle de qualidade no Netlogo.	134
Listagem 7	Código para implementação da mudança de produto no Netlogo.	136
Listagem 8	Código para monitoração de <i>Work In Process</i> e <i>Manufacturing Lead</i> <i>Time</i>	138
Listagem 9	Verificação do tempo de execução em WorkStation.	139
Listagem 10	Fragmento de arquivo do programa de controle da DWS.	142
Listagem 11	Código completo para implementação do modelo Netlogo apresentado no Capítulo 5.	164

LISTA DE ABREVIATURAS E SIGLAS

AC	<i>Asynchronous Channel</i>
ACL	<i>Agent Communication Language</i>
ADACOR	<i>ADaptive holonic COntrol aRchitecture</i>
AWS	<i>Assembling WorkStation</i>
BPM	<i>Business Process Management</i>
CA	<i>Controller Agent</i>
CFA	<i>ConFiguration Agent</i>
CIM	<i>Computer-Integrated Manufacturing</i>
CLP	<i>Controlador Lógico Programável</i>
CNC	<i>Controle Numérico Computadorizado</i>
CNP	<i>Contract Net Protocol</i>
DWS	<i>Distributing WorkStation</i>
FIPA	<i>Foundation for Intelligent Physical Agents</i>
FPGA	<i>Field Programmable Gate Array</i>
GALS	<i>Globally-Asynchronous Locally-Synchronous</i>
GUI	<i>Graphical User Interface</i>
HMAS	<i>Holonic and Multi-Agent System</i>
IOPT	<i>Input Output Place Transition</i>
IWS	<i>Inspection WorkStation</i>
JADE	<i>Java Agent DEvelopment framework</i>
MLT	<i>Manufacturing Lead Time</i>
MPS	<i>Modular Production System</i>
MTBF	<i>Mean Time Between Failures</i>
OpH	<i>Operational Holon</i>
PFS	<i>Production Flow Schema</i>
PWS	<i>Processing WorkStation</i>
PrH	<i>Product Holon</i>
PROSA	<i>Product Resource Order Staff Architecture</i>
QCWS	<i>Quality Control WorkStation</i>
RMT	<i>Reconfigurable Machine Tools</i>
RdP L/T	<i>Rede de Petri Lugar/Transição</i>
RdP	<i>Rede de Petri</i>
SCADA	<i>Supervisory Control And Data Acquisition</i>

SED	Sistema a Eventos Discretos
SFC	<i>Sequential Flow Chart</i>
SOA	<i>Service-Oriented Architecture</i>
SRM	Sistema Reconfigurável de Manufatura
SuH	<i>Supervisor Holon</i>
TrWS	<i>Transporting WorkStation</i>
TeWS	<i>Testing WorkStation</i>
TH	<i>Task Holon</i>
WADE	<i>Workflow Agent Development Environment</i>
WEA	<i>Workflow Engine Agent</i>
WIP	<i>Work In Process</i>
WS	<i>WorkStation</i>
WebS	<i>Web Services</i>
WSDL	<i>Web Services Description Language</i>
XML	<i>eXtensible Markup Language</i>
XPDL	<i>XML Process Definition Language</i>

LISTA DE SÍMBOLOS

P	conjunto de lugares da RdP
T	conjunto de transições da RdP
F	conjunto de arcos orientados da RdP
W	conjunto de pesos de cada arco da RdP
M	marcação da RdP
p	lugar da RdP
m_i	número de marcas do lugar “i” da RdP
$[c_{ij}]$	matriz de incidência da RdP
TA	conjunto de arcos teste da IOPT
$WTest$	função peso associada aos arcos testes na IOPT
isg	função de input signal guard na IOPT
BE	conjunto finito de funções booleanas na IOPT
IS	conjunto finito de input signals na IOPT
ie	função de input event associado a transições na IOPT
IE	conjunto finito de input events na IOPT
oe	função de output events associado a transições na IOPT
OE	conjunto finito de output events na IOPT
osc	função de output signals na IOPT
$Var(e)$	função que retorna o conjunto de variáveis em uma dada expressão
ML	conjunto de identificadores para cada marcação de lugar na IOPT
td	time domain
AC	Asynchronous Channel
τ_{due}	prazo devido para que determinado processo complete suas operações
$\Delta\tau_{due}$	intervalo de tempo em que cada tarefa deve completar suas operações
$task$	tarefa de determinado processo produtivo
H	holarquia
h	holon
C_{H_n}	variável custo relacionada certa holarquia n
C_{task_n}	variável custo relacionada a certa tarefa n de um holon
π	fundo do serviço
φ	penalidade pelo atraso de certo serviço
ε	recompensa pela realização de certo serviço
μ	variável relacionada a desempenho dos holons
α	fator de autonomia

$\Delta\tau_r$	intervalo de tempo estimado para recuperação da falha
ρ	parâmetro do tipo de falha

SUMÁRIO

1 INTRODUÇÃO	20
1.1 OBJETIVO	28
1.2 ORGANIZAÇÃO DO TEXTO	29
2 REVISÃO BIBLIOGRÁFICA	30
2.1 SISTEMAS RECONFIGURÁVEIS DE MANUFATURA	30
2.2 REDE DE PETRI	40
2.2.1 Production Flow Schema	41
2.2.2 Rede de Petri Lugar/Transição (RdP L/T)	42
2.2.3 Input Output Place Transition	45
2.3 HOLONS E AGENTES	48
2.3.1 Análise de sistemas multiagentes	51
2.3.2 Projeto e implementação de sistemas multiagentes	54
2.3.3 Ontologia para HMAS	58
2.3.4 Outros trabalhos correlatos	60
2.4 ARQUITETURA ORIENTADA A SERVIÇOS	63
2.5 DISCUSSÃO	65
3 ARQUITETURA DE CONTROLE PROPOSTA	69
3.1 WORKFLOW PARA A CONCEPÇÃO DA ARQUITETURA	69
3.2 COMPOSIÇÃO DAS CLASSES	70
3.3 MECANISMOS DE TRATAMENTO DE FALHAS	74
3.4 MECANISMOS PARA COMPOSIÇÃO DE HOLARQUIAS	76
3.5 MECANISMOS PARA RECONFIGURAÇÃO	78
3.6 SÍNTESE DO CAPÍTULO	82
4 IMPLEMENTAÇÃO DA ARQUITETURA DE CONTROLE	84
4.1 MODELAGEM CONCEITUAL	86
4.1.1 Definição da Ontologia	86
4.1.2 Modelagem conceitual dos holons	88
4.1.3 Orquestração dos modelos conceituais	93
4.2 MODELAGEM FUNCIONAL	94
4.2.1 Modelagem dos holons em IOPT	96
4.2.2 Orquestração dos modelos funcionais	101
4.2.3 Modelagem de mecanismos de diagnóstico e tomada de decisão	105

4.2.4 Análise e Revisão dos Modelos	107
4.3 IMPLEMENTAÇÃO	109
4.4 SÍNTESE DO CAPÍTULO	110
5 EXEMPLO DE APLICAÇÃO	113
5.1 MODELAGEM CONCEITUAL	117
5.2 MODELAGEM FUNCIONAL	120
5.3 IMPLEMENTAÇÃO	139
5.4 OBSERVAÇÕES ADICIONAIS	140
6 CONCLUSÕES	146
6.1 TRABALHOS FUTUROS	148
Referências	150
Apêndice A – Código para implementação do modelo Netlogo	164
Apêndice B – Publicações Decorrentes do Trabalho	202

1 INTRODUÇÃO

Os sistemas de manufatura são concebidos para atender as demandas de produção, isto é, para transformar material e/ou processar informações na execução de uma ordem de produção. Um sistema de manufatura é composto por vários subsistemas que executam diferentes processos produtivos que por sua vez interagem de diferentes formas. Um subsistema de um sistema de manufatura consiste de vários componentes tais como máquinas de produção, sistema de manuseio de material, sistemas computacionais e operadores humanos (GROOVER, 2007; KOREN; SHPITALNI, 2010; LEITÃO et al., 2015).

A competitividade crescente no mercado globalizado e a necessidade de eficiência têm imposto mudanças nos sistemas de manufatura exigindo maior flexibilidade sob diferentes aspectos, tais como o volume de produção, o tipo de produto e/ou serviço e a natureza dos recursos envolvidos. Para considerar diferentes requisitos, os sistemas de manufatura são organizados de diferentes formas. Mehrabi, Ulsoy e Koren (2000) relacionam estas formas de organização com os objetivos econômicos impostos pelo mercado para obter vantagens competitivas: (i) sistema de manufatura em massa para obter redução de custos de produto, (ii) sistema de manufatura enxuta para prover a qualidade do produto, (iii) sistema de manufatura flexível para aumento da variedade de produtos, e (iv) Sistema Reconfigurável de Manufatura (SRM) para assegurar a agilidade de adaptar o sistema às mudanças da planta industrial. Outras formas de organização consideram uma combinação dos objetivos citados, tais como o sistema de manufatura para a personalização em massa que considera a flexibilidade de oferecer uma vasta quantidade de famílias de produtos individualizados de acordo com as necessidades do cliente (TSENG; HU, 2014).

A quarta revolução industrial, chamada de “Indústria 4.0”, é considerada segundo alguns pesquisadores, o futuro padrão de sistemas de produção (KOLBERG; ZÜHLKE, 2015). Na Indústria 4.0, novas tecnologias devem ser empregadas para integrar máquinas e humanos em cadeias de valor compondo uma rede de entidades (onde estas podem ser plantas industriais geograficamente distribuídas e dispersas) que devem fornecer produtos e/ou serviços de forma autônoma, atendendo os objetivos globais do sistema produtivo. Ou seja, observa-se que os sistemas de manufatura podem ser dispersos, sendo composto por vários subsistemas em localizações geográficas diferentes que fornecem serviços específicos com certo grau de autonomia e com comunicação baseada na internet, onde cada subsistema encapsula um serviço (MELO et al., 2008; FATTORI et al., 2011; NAGORNY; COLOMBO; SCHMIDTMANN, 2012). A Indústria 4.0 também envolve o uso de dispositivos do tipo *smart* no chão de fábrica (SCHMITT; ORFGEN; ZÜHLKE, 2015), permitindo o acesso informações detalhadas de cada produto e com isto maior interação entre clientes e fornecedores para

atender as demandas individuais. Outro requisito da Indústria 4.0 é a necessidade dos sistemas de controle de suportar reconfiguração *plug-and-play* dos seus componentes (KOLBERG; ZÜHLKE, 2015). Esta reconfiguração é necessária para adição, modificação e remoção de recursos (isto é, máquinas, operadores, etc.) ou para alteração de alguma parte de um fluxo do processo produtivo. Portanto, a Indústria 4.0 mostra-se como uma efetiva abordagem para se contemplar os objetivos dos sistemas de produção citados no parágrafo anterior, isto é, sistemas de manufatura com capacidade de reconfiguração com integração dos avanços tecnológicos relacionados a sistemas cibernéticos (LEITÃO et al., 2015) e aplicação de conceitos relacionados a sistemas distribuídos, explorando o potencial de computação nas nuvens (LI et al., 2010).

Apesar das vantagens das propostas de novas formas de organização de sistemas de manufatura, para que o potencial das novas e heterogêneas tecnologias seja integrado, é necessário rever os métodos¹ tradicionais de concepção e análise de seus sistemas de controle considerando os aspectos em questão da sua organização. Assim, existem diversas soluções de engenharia que procuram facilitar o desenvolvimento de aplicações dedicadas ao controle e automação para problemas específicos dos subsistemas de um sistema de manufatura (STRASSER et al., 2009; MOREL et al., 2007). Entretanto, ainda não se tem uma resposta para a integração destas soluções que, na prática, envolve contemplar em um sistema de controle, diferentes visões/abordagens dos vários domínios existentes, em um tempo relativamente curto devido à concorrência/dinâmica entre as empresas.

A definição de um sistema de controle depende essencialmente de uma arquitetura de controle e de um método que permita especificar as funcionalidades e implementar de fato os componentes desta arquitetura (DA SILVA et al., 2016). Esta especificação, por sua vez, consiste em detalhar tanto a estrutura como as funções dos componentes do sistema de controle, suas interfaces, inter-relações e restrições. Em geral, o sistema de manufatura adota uma estrutura hierárquica na qual redes de comunicação, tais como LAN e *fieldbus* são fundamentais para implementar as malhas de comunicação para troca de mensagens entre os dispositivos. De acordo com normas internacionais como ANSI/ISA (1995), a organização da estrutura de decisões dos sistemas convencionais de manufatura é formada pelas seguintes camadas: nível corporativo, nível de fábrica ou produção, nível de controle supervisão, nível de máquina e nível de dispositivo.

Sob outro ponto de vista, isto é, sob a abordagem de teoria de sistemas, um sistema de manufatura pode ser classificado como um Sistema a Eventos Discretos (SED), que é um sistema dinâmico que evolui de acordo com a ocorrência abrupta de eventos, que podem ocorrer

¹Modo de proceder para atingir certo objetivo (FERREIRA, 2016).

em intervalos irregulares de tempo possivelmente desconhecidos (RAMADGE; WONHAM, 1989; CASSANDRAS; LAFORTUNE, 2008). No controle de SED, ou controle discreto, os parâmetros e as variáveis dos processos produtivos assumem valores discretos (normalmente binários, tais como “ligado/desligado”) que são modificados em momentos que também são discretos no tempo.

Em relação aos aspectos que caracterizam os sistemas feitos pelo homem e sua propriedade intrínseca de ocorrência de falha, para assegurar que os processos produtivos do sistema de manufatura não sofram interrupções, Zhang e Jiang (2008) propõem o sistema de controle ativo e tolerante a falhas. Este sistema possui mecanismos para a detecção da falha, o estudo dos seus efeitos, identificação das causas e a reconfiguração do sistema que é feita realocando funções e escolhendo caminhos alternativos de interação entre os processos (ARAKAKI; MIYAGI, 2006).

Considerando a realidade em que os tipos de produto/processos têm reduzidos ciclos de vida, uma alternativa é reutilizar/reconfigurar componentes do sistema original em novos sistemas de manufatura. De acordo com Koren e Shpitalni (2010), o SRM deve permitir a produção de grande quantidade de uma família de produtos personalizados, considerando sistemas de melhoria contínua de qualidade, flexibilidade e ágil adaptabilidade às mudanças da planta industrial, de tipo/quantidade de produtos, processos, recursos e, além disso, deve assegurar a devida reação à ocorrência de falhas. Além disso, um SRM deve ser responsivo, ou seja, deve permitir a reconfiguração em funcionalidade e capacidade de produção adaptando-se às mudanças de mercado, produto e recurso. De fato, os fluxos de trabalho em uma empresa estão sujeitos a alterações devido às mudanças de processos de negócios, ambiente operacional, recursos disponíveis e necessidades do cliente.

Neste contexto, os sistemas de controle que têm a capacidade de reconfigurar a dinâmica de produção de itens devem atender novas demandas com agilidade, flexibilidade e aumento de eficiência. O termo “novas demandas” pode se referir a: novos produtos, mudanças de quantidade de produtos, integração de novas tecnologias/processos ou situações causadas por ocorrência de falhas. Além disso, considerando o mercado globalizado e a rede de entidades geograficamente dispersas que fazem parte dos sistemas produtivos, entende-se que o controle de um SRM deve considerar as características de sistemas dispersos, tais como *service cloud manufacturing* (DA SILVA et al., 2014), isto é, uso de *hardware* e *software* interligados por rede de comunicação com acesso remoto (NAGORNY; COLOMBO; SCHMIDTMANN, 2012), onde atividades operacionais do sistema de manufatura são descritas como serviços.

Os sistemas de controle que consideram responsividade devem cobrir as características de: personalização (capacidade de personalizar alguns produtos), convertibilidade (capacidade de mudar a funcionalidade dos subsistemas), escalabilidade (capacidade de modificar a produção,

adicionando/removendo recursos/componentes), modularidade (capacidade de representar o sistema em módulos de controle, permitindo o reuso e a modelagem de sistemas complexos), integrabilidade (capacidade de integrar com agilidade e eficiência dispositivos, informações e interfaces de controle) e diagnosticabilidade (capacidade de detectar e diagnosticar o estado do sistema).

Em uma estrutura de sistema disperso, o controle de um SRM deve assim ser distribuído com características de autonomia, inteligência, tolerância às situações imprevistas e facilidade de reuso de seus componentes, além de flexibilidade e agilidade. Como esse sistema de controle deve ser distribuído, a ausência de uma coordenação hierárquica requer que cada componente tenha sua própria capacidade de tomada de decisão, o que em muitos casos inviabilizaria sua implementação prática (PUJO; BROISSIN; OUNNAR, 2009). Neste sentido, do ponto de vista de uma arquitetura de controle de um sistema reconfigurável de manufatura, é necessária uma revisão da sua estrutura organizacional, pois os processos produtivos não são estáticos e o sistema precisa responder às mudanças com agilidade e considerando inclusive a possibilidade de falhas na rede de comunicação que integra os diferentes elementos do sistema de controle. Além disso, esta arquitetura de controle determina os processos de decisão.

Um método de modelagem eficaz deve permitir a especificação da solução de controle do SRM capaz de acomodar as características acima citadas. Por exemplo, quando um evento inesperado ocorre e uma reconfiguração é necessária, os componentes envolvidos no sistema de controle devem interagir considerando os diferentes sistemas operacionais e tecnologias de cada subsistema (HSIEH, 2010). Além disso, o método em questão deve adequar protocolos para sistematizar a comunicação entre diferentes equipes envolvidas no projeto deste sistema de controle. Entretanto, apesar da ideia de novos paradigmas ser atrativa, a definição deste método não é trivial. A complexidade deste tipo de sistema aliada à falta de medições quantitativas das vantagens das novas tecnologias, restringem a adoção de abordagens existentes (HSIEH; LIN, 2013; VRBA et al., 2011; COLOMBO; NEUBERT; SCHOOP, 2001; PĚCHOUČEK; MAŘÍK, 2008).

Quanto ao aspecto de complexidade, a modelagem de processos do SRM necessita abordar técnicas de modularização e com formalismos que permitam a aplicação/análise sistemática dos conceitos envolvidos. Por sua vez, para viabilizar a implementação prática de SRM é fundamental considerar a reutilização de modelos, sua interoperabilidade e portabilidade (TOMMILA et al., 2005). Desse modo, busca-se evitar a repetição de tarefas com sobreposição de escopos na especificação dos “domínios”². Neste sentido, a adoção do conceito de

²Problemas para os quais uma determinada aplicação pretende dar solução (BELLIFEMINE; POGGI; RIMASSA, 1999).

componente³ permite a definição de uma linguagem comum e o reuso de modelos. Em SRM, cada componente da arquitetura de controle deve ser autônomo com conhecimento, inteligência e habilidade necessários para atingir o seu objetivo e interagir com outros componentes para atingir o objetivo global do sistema.

Mick e Polsonetti (2003), Harrison e Colombo (2005) e Colombo et al. (2004) propõem assim uma nova classe de sistemas com componentes distribuídos que colaboram para executar as atividades de controle, denominada “automação colaborativa”. Segundo Colombo e Harrison (2008), para implementar a automação colaborativa são necessários novos projetos de máquinas e sistemas de controle, que devem migrar de sistemas legados⁴ para sistemas baseado em módulos colaborativos. A modularização do SRM requer a revisão do conceito em geral adotado nas empresas de *controller-oriented structure* em módulos funcionais baseados no conceito de *manufacturing-task-oriented structure* (COLOMBO; HARRISON, 2008). Esta solução deve propiciar sistemas mais ágeis pois respostas rápidas são fundamentais para atender as mudanças de requisitos dos clientes e do ambiente de mercado. Esta colaboração em SRM faz com que empresas, clientes e parceiros trabalhem juntos para obter maior ganho.

Portanto, métodos para especificação do controle de SRMs devem considerar características como:

- (i) responsividade, considerando a reconfiguração em funcionalidade e em produção de acordo com as mudanças de mercado, produtos e recursos (KOREN; SHPITALNI, 2010). A reconfiguração da funcionalidade permite criar novos sistemas, atualizar e/ou substituir sistemas legados e modificar processos no caso de novas demandas (MEHRABI; ULSOY; KOREN, 2000; ASATO et al., 2015). A reconfiguração da produção envolve a possibilidade de alteração da quantidade de produtos pré-definidos (KOREN; SHPITALNI, 2010). Além disso, falhas são inevitáveis em sistemas concebidos e operados pelo homem (MIYAGI; RIASCOS, 2006) e os mecanismos de reconfiguração devem recuperar o funcionamento do sistema no caso de ocorrência de alguma falha. Ou seja, além de atender aos requisitos citados, o método para a concepção e análise dos sistemas de controle deve explorar as relações de colaboração entre as partes envolvidas através de mecanismos que permitam tanto atender novas demandas de produtos/processos como resolver situações de ocorrência de falha;

³Neste caso, componente refere-se a um bloco que encapsula *hardware*, *software*, estrutura de dados e algoritmos (TOMMILA et al., 2005).

⁴Sistema composto por subsistemas que, apesar de terem recursos (máquinas, equipamentos, etc.) e processos produtivos considerados obsoletos, fornecem serviços essenciais do sistema de manufatura (JENNINGS, 2001).

- (ii) autonomia e inteligência de seus componentes (MAŘÍK; LAŽANSKÝ, 2007) considerando a distribuição física de sistemas dispersos. A solução de controle de um sistema de manufatura deve integrar *hardware* e *software* dos subsistemas, os quais podem estar organizados em diferentes estruturas e especificações heterogêneas de diferentes tecnologias. Como exemplo de equipamentos com diferentes *hardware* e *software* tem-se: robô, máquinas de Controle Numérico Computadorizado (CNC), *Supervisory Control And Data Acquisition* (SCADA), *Field Programmable Gate Array* (FPGA) e Controlador Lógico Programável (CLP);
- (iii) sistemas com componentes distribuídos que colaboram para executar as atividades de controle. Estes componentes devem ser tratados segundo o conceito de módulos para permitir a reutilização, definição de interfaces para sua interoperabilidade e portabilidade;
- (iv) supervisão global de todos os processos que compõem o sistema combinado com a autonomia distribuída existente (GIRET; TRENTESAUX, 2015), para uso integrado dos recursos tecnológicos/humanos e agilidade na reação a ocorrência de falhas/mudanças;
- (v) integração entre homem e máquina em cadeias de valor compondo uma rede de plantas industriais localizadas em posições geograficamente dispersas e distribuídas, onde atividades operacionais do sistema de manufatura são descritas como serviços (NAGORNY; COLOMBO; SCHMIDTMANN, 2012; KOLBERG; ZÜHLKE, 2015).

Mařík et al. (2015) afirmam que uma tendência que abrange as características requeridas para a estrutura de controle de sistema reconfigurável de manufatura, e que também considera aspectos da visão da Indústria 4.0, é a arquitetura baseada em “holons” (KOESTLER, 1968) e “agentes” (FERBER, 1999; JENNINGS; BUSSMANN, 2003; COLOMBO; NEUBERT; SCHOOP, 2001). Nesta arquitetura, cada recurso de um sistema pode ser representado como um “agente de produção”, que contém todos os parâmetros necessários para caracterizar funcionalmente os seus dispositivos mecatrônicos. Além disso, o *software* de controle e supervisão baseado em agentes pode ser dotado com capacidades de comunicação e processamento de informação, transformando-o num elemento autônomo e inteligente, isto é, um holon. A ideia é que se possa explorar os conceitos de superposição de sistemas multiagentes e sistemas holônicos, tais como autonomia, reatividade, proatividade, cooperação, interface social (isto é, consideração da interação humana nos processos produtivos), mecanismos de aprendizagem, e tirar proveito das vantagens características da implementação de sistemas holônicos por meio de sistemas multiagentes (GIRET; BOTTI, 2004). PROSA (Van Brussel et al., 1998), MetaMorph (MATURANA; SHEN; NORRIE, 1999) e ADACOR (LEITÃO; RESTIVO, 2006) são alguns exemplos de arquitetura de controle baseado nos

conceitos de *Holonic and Multi-Agent System* (HMAS). Entretanto, segundo Vrba et al. (2011); Colombo, Neubert e Schoop (2001); Pěchouček e Mařík (2008), existe um longo caminho para disseminação e divulgação desta tecnologia e a adoção da nova arquitetura pois, isso depende do desenvolvimento de novos métodos e ferramentas que facilitem a migração das abordagens tradicionais para novas soluções. De acordo com Hsieh (2009), um desafio desta nova arquitetura é o mecanismo para que o HMAS atinja os objetivos de forma autônoma necessariamente dentro de um prazo requerido. De acordo com (OBITKO et al., 2010), outro desafio é integrar o HMAS com a definição de modelos semânticos aplicados a sistemas de manufatura, permitindo assim maior aceitação desta técnica no âmbito industrial. Além disso, em geral, as abordagens baseadas em HMAS não consideram aspectos da interoperabilidade para comunicação entre sistemas dispersos.

A *Service-Oriented Architecture* (SOA) foi concebida para tratar a colaboração entre empreendimentos fornecendo padrões para interfaces. Além disso, entende-se neste trabalho que existe uma sinergia entre SOA e HMAS que pode ser explorada. Por exemplo, agentes podem fornecer serviços para outros agentes do mesmo modo como serviços são fornecidos em SOA e ambas tecnologias são baseadas em troca de mensagens entre seus componentes (SHEN et al., 2007). Estudos realizados nesta área (BRENNAN; FLETCHER; NORRIE, 2002; CHIRN; MCFARLANE, 2000; CHRISTENSEN, 2003; COLOMBO; NEUBERT; SCHOOP, 2001; HSIEH, 2009; LEITÃO; RESTIVO, 2006; PARUNAK; BAKER; CLARK, 2001; VALCKENAERS et al., 1994; VRBA et al., 2011), demonstram que, em geral, o que se tem é a indicação de padrões sem uma técnica sistemática que facilite e torne mais compreensível o desenvolvimento dos modelos a serem utilizados na avaliação funcional e implementação de um sistema reconfigurável de manufatura.

Voltando a visão de teoria de sistemas, em relação às técnicas de modelagem de processos, segundo Reisig (1985), a Rede de Petri (RdP), se comparada com outras técnicas de descrição de SED, tais como cadeias de Markov, teoria de filas, álgebra de processo, álgebra *max-plus*, lógica temporal, teoria de linguagem e/ou autômatos, possui um poder equivalente de modelagem. Entretanto, tem a característica e vantagem da facilidade de visualização do sistema em diferentes níveis de abstração (VILLANI; MIYAGI; VALETTE, 2007; MIYAGI, 1996; LEITÃO; COLOMBO, 2006; FELDMANN et al., 1999). Uma vantagem adicional é que algumas classes de RdP, tais como a *Input Output Place Transition* (IOPT) (GOMES et al., 2007), pode ser convertida para programas de controladores industriais utilizando um *software* aberto associado à proposta que está disponível na internet (GOMES et al., 2007). Outro tipo de RdP é o *Production Flow Schema* (PFS) (HASEGAWA et al., 1999) que pode ser explorado como um *workflow* tanto para comunicação entre equipes que trabalham com diferentes tecnologias, informação, recursos e conhecimento, como para especificação do fluxo

entre os componentes de controle de um sistema reconfigurável de manufatura. De acordo com Caire, Gotta e Banzi (2008), o uso de ferramentas baseadas em *workflow* facilita o desenvolvimento de programas computacionais e de controle, pois é mais compreensível por especialistas no domínio da aplicação, independente de seu conhecimento em programação. Alguns trabalhos aplicam a combinação das técnicas de RdP (tais como o PFS e RdP do tipo lugar transição) em uma abordagem *top down* (ASATO et al., 2011). No entanto, para um sistema reconfigurável de manufatura, devido suas específicas características de distribuição, autonomia, inteligência, etc., esta abordagem não é suficiente para garantir um *framework* para validar estratégias de controle.

Portanto, no desenvolvimento do controle de SRMs, analisando-se os trabalhos acima citados, as técnicas SOA, HMAS, RdP e outras boas práticas de projeto (tais como o uso de ferramentas disponíveis de código aberto e que sejam baseadas em *workflow*), identificam-se vários aspectos que poderiam ser devidamente combinados. Nesta solução, os desafios identificados são:

- (i) o desenvolvimento de mecanismos para a composição de holarquias para garantir a autonomia de HMAS, a execução de processos produtivos considerando restrições temporais e a capacidade/disponibilidade de processamento da máquina. Os mecanismos devem ser baseados na funcionalidade do equipamento, pois quando um processo solicita um recurso, na verdade ele está solicitando a funcionalidade e a capacidade mais adequada para executar uma ordem de produção;
- (ii) a adequação de protocolos e sistematização de implementação e/ou atualização dos projetos do sistemas de controle considerando a definição de uma semântica desde a concepção e especificação das soluções de controle até a efetiva programação dos controladores industriais;
- (iii) a definição da arquitetura de controle e supervisão que integre HMAS e SOA aplicado a um sistema reconfigurável de manufatura. Nesta integração é fundamental que holons de baixo nível de controle sejam parte de alguns holons de alto nível colaborando com outros holons e com certo grau de autonomia; e
- (iv) a especificação de estratégias de controle e supervisão que permitam a reconfiguração de novas demandas de produtos/processos como resolver situações de ocorrência de falha.

1.1 OBJETIVO

Este trabalho visa a concepção de uma solução de controle para sistemas reconfiguráveis de manufatura o qual contempla os desafios identificados anteriormente, definindo sua arquitetura de controle e um método para implementação prática desta arquitetura.

De forma específica, são consideradas as seguintes metas em relação à arquitetura de controle:

- (i) identificação de conceitos e técnicas de SRMs, modelagem e implementação de HMAS, SOA e RdP;
- (ii) geração de uma especificação documentada por modelos baseada em componentes distribuídos que facilitem seu reuso. Os modelos devem encapsular dados e processos, além de uma definição semântica;
- (iii) realização da decomposição física e funcional do sistema de manufatura. A decomposição física envolve a identificação de componentes, tais como operadores, ferramentas, programação, produtos, ordens, atributos e operações. A decomposição funcional considera funcionalidades, tais como aquisição de ordem de trabalho, planejamento, programação, manipulação de material, logística, etc.;
- (iv) virtualização das habilidades de manufatura como serviço e disponibilização a consumidores por meio de plataformas HMAS considerando o ambiente de computação nas nuvens. Ou seja, transformação da estratégia de controle em recurso de fábrica ou função de fábrica para componentes autônomos que podem encapsular e fornecer a consumidores suas funcionalidades e habilidades como serviços;
- (v) desenvolvimento de mecanismos de reconfiguração que permitam tanto atender novas demandas de produtos/processos como resolver situações de ocorrência de falha; e
- (vi) integração dos diferentes níveis do processo de negócios de um sistema de manufatura, desde a ordem de cliente, produção, até os processos de *marketing*.

Em relação ao desenvolvimento do método para implementação da arquitetura de controle, deve-se levar em conta as metas:

- (i) desenvolvimento de procedimento de modelagem de fácil compreensão porém com formalismo bem definido;
- (ii) consideração de disponibilidade de ferramentas de análise baseada em simulação de cenários e verificação dos modelos;

- (iii) consideração da disponibilidade de técnicas para geração de códigos para a implementação automática dos modelos; e
- (iv) consideração de níveis de abstração, fornecendo artefatos de modelagem/ferramentas e procedimentos para gerenciar o processo.

Em relação à avaliação da viabilidade do sistema de controle proposto e análise dos resultados alcançados, o trabalho envolve a meta de implementar estudos de caso de SRMs.

1.2 ORGANIZAÇÃO DO TEXTO

O Capítulo 2 descreve o levantamento bibliográfico das técnicas, conceitos, trabalhos relacionados à modelagem e descrição do controle de sistemas reconfiguráveis de manufatura. Especial atenção foi atribuída para HMAS, técnicas de modelagem de sistemas como Rede de Petri Lugar/Transição (RdP L/T), *Input Output Place Transition* (IOPT) e *Production Flow Schema* (PFS), mecanismos de controle tolerante à falhas e métodos de projeto. As principais vantagens e desvantagens de trabalhos correlatos são também descritas. Ou seja, este capítulo descreve os conceitos fundamentais a serem considerados para a concepção de uma solução de controle para SRMs e características necessárias para projeto do sistema de controle de um SRM, isto é: sistema distribuído, autonomia, inteligência, tolerância a falhas, adaptabilidade e reuso de modelos. Também são consideradas as técnicas formais para modelagem e análise de sistemas de controle. Uma discussão ao final deste capítulo destaca as características requeridas para projetar o controle de um SRM e as ferramentas disponíveis para a implementação de seu sistema de controle.

Baseado no levantamento anterior, o Capítulo 3 apresenta uma visão geral do *workflow* proposto para a concepção da arquitetura de controle de sistemas reconfiguráveis de manufatura, descrevendo seus componentes e as técnicas envolvidas. Na sequência deste mesmo capítulo, são apresentados os mecanismos adotados e propostos para o controle de sistema reconfigurável de manufatura, composição de holarquias e de tratamento de falhas. Algumas ferramentas são selecionadas para utilização na implementação da arquitetura.

O Capítulo 4 apresenta o método proposto e seus procedimentos para obter os componentes da arquitetura de controle, utilizando a combinação de técnicas de SOA, HMAS e RdP. O método permite evoluir da definição do problema até a sua implementação apresentada no Capítulo 5 onde um exemplo ilustra uma aplicação, a qual usa ferramentas já disponíveis para edição, depuração, simulação e geração de programas de controladores. O Capítulo 6 apresenta as principais conclusões e contribuições deste trabalho.

2 REVISÃO BIBLIOGRÁFICA

2.1 SISTEMAS RECONFIGURÁVEIS DE MANUFATURA

De acordo com Ali, Seifoddini e Sun (2005) e Groover (2007), o sistema de manufatura transforma insumos (*inputs*) para obtenção de um produto final (*output*) agregando valor de mercado. Kosanke, Vernadat e Zelm (1999) confirmam que além do sistema de manufatura ter o objetivo de obter um produto final, detalham que este é composto por produtos intermediários obtidos a partir de execução de operações de processos produtivos em matérias-primas.

Para atender as necessidades do mercado explorando o desenvolvimento de novas tecnologias, os sistemas de manufatura evoluíram e são organizados de formas diferentes. De acordo com Mehrabi, Ulsoy e Koren (2000), um SRM deve ser projetado para ajustar rapidamente a capacidade de produção, em resposta a novas demandas, por meio de rearranjo ou mudança de componentes, que podem ser máquinas, esteiras, dispositivos e programas de controladores. Um SRM abrange as características de outras formas de organização de sistemas de manufatura e adota a habilidade de ajustar a capacidade e funcionalidade da produção a mudanças imprevistas e novas demandas por meio da reconfiguração do sistema. Novas demandas podem ser um novo produto, quantidade de produtos ou integração de novas tecnologias/processos. Esta ideia de demanda pode ser estendida para o caso de novos processos, visto que os fluxos de trabalho em uma empresa estão sujeitos a alterações devido às mudanças de processos de negócios, ambiente operacional, recursos e necessidades do cliente.

Koren e Shpitalni (2010) descrevem características de SRMs. Segundo os autores, um SRM deve considerar a “responsividade” que é relacionado à velocidade de reação da planta às mudanças de mercado, as quais incluem mudanças na quantidade de produto; de produtos, as quais incluem mudanças do produto atual do sistema ou relacionadas à introdução de novos produtos; e à ocorrência de falhas. Ou seja, o SRM permite ajustar a capacidade de produção conforme a demanda de mercado e de produto mudam. Para tanto, dois tipos de reconfiguração são necessárias para os sistemas de manufatura: reconfiguração em funcionalidade e em capacidade de produção. Figura 1 ilustra a comparação da reação entre os sistemas de manufatura para um exemplo onde um produto “C” diferente de “A” e “B” que foram concebidos no projeto inicial do sistema de manufatura, é requerido e onde a demanda de produtos também varia de forma diferente do planejado.

De acordo com Putnik et al. (2013) e Wang e Koren (2012), para que um sistema de manufatura possa reagir a variações de demanda, o mesmo deve ser concebido considerando escalabilidade que é a habilidade do sistema de manufatura mudar sua capacidade de produção.

Figura 1 - Exemplo de mudanças de demanda e tipo de produto para sistemas de manufatura.

Adaptado de Koren e Shpitalni (2010).

Embora alguns sistemas flexíveis de manufatura sejam concebidos para produzir uma ampla variedade de produtos, em geral, eles não são projetados para mudanças de demanda e com mecanismos para reação a falhas (MOSLEMIPOUR; LEE; RILLING, 2012). Em geral, o sistema flexível de manufatura envolve a concepção de um sistema com máquinas CNC e outros equipamentos mecatrônicos para a produção de grande variedade pré-concebida de produtos. Entretanto, exatamente por não adotar equipamentos dedicados a um produto específico, os sistemas flexíveis de manufatura têm uma taxa de produção bem menor do que sistemas de produção em massa. Além disso, a combinação de alto custo por equipamento com a relativamente reduzida taxa de produção torna o custo por produtos em sistemas flexíveis de manufatura relativamente alto (PUTNIK et al., 2013). De acordo com Koren e Shpitalni (2010), um sistema de manufatura é reconfigurável se a resposta é positiva para:

- (i) O sistema de manufatura e seus equipamentos foram projetados de forma que sua estrutura física pode ser facilmente modificado?
- (ii) O sistema de manufatura e seus equipamentos foram projetados para produção e inspeção de uma certa família de produtos?

Por exemplo, se um sistema de manufatura fabrica pistões mecânicos de simples ação, dependendo do tipo da corpo do pistão (base cilíndrica) e dos componentes internos (pino, mola e tampa) vários produtos finais podem ser obtidos dependendo da combinação de montagem e das operações de usinagens, furação, prensa, etc. que são realizadas em cada produto intermediário. A visão de uma família de produto permite prever diferentes variações de produto com poucas variações no processo de produção. Desse modo, SRMs combinam a alta produtividade de sistemas de produção em massa com soluções com menores custos que sistemas flexíveis de manufatura.

Koren e Shpitalni (2010) apresentam um comparativo, conforme listado na Tabela 1, com os sistemas de produção em massa e os sistemas flexíveis de manufatura para demonstrar as vantagens/características do Sistema Reconfigurável de Manufatura.

Tabela 1 - Comparativo entre sistemas de manufatura: reconfigurável, dedicado e flexível

	dedicado	reconfigurável	flexível
estrutura do sistema	fixa	modificável	modificável
estrutura da máquina	fixa	modificável	fixa
foco do sistema	produto	família de produtos	máquina
escalabilidade	não	sim	sim
flexibilidade	não	personalizado	geral
operação simultânea	sim	possível	não
produtividade	muito alta	alta	baixa
custo por produto	baixo (se completamente utilizado)	médio	razoável

Adaptado de Koren e Shpitalni (2010).

De acordo com Koren (2014), SRMs devem envolver seis características fundamentais:

- (i) “personalização”, relacionada à possibilidade de produzir certa quantidade de produtos (isto é, limitada a uma família de produtos) através da adoção de técnicas de flexibilidade do sistema ou de máquina;
- (ii) “convertibilidade”, relacionada à capacidade de mudar funcionalidades do sistema para adequar-se a novos requisitos de produção;
- (iii) “escalabilidade”, relacionada à capacidade de mudar a taxa de produção do sistema seja adicionando, substituindo ou removendo recursos e/ou mudando processos;
- (iv) “modularidade”, relacionada à descrição/organização de funções em unidades operacionais que podem ser modificadas para formar diferentes esquemas de produção;
- (v) “integrabilidade”, relacionada à definição de interfaces mecânicas, de dados e de controle, que facilitam a comunicação e integração dos módulos de forma precisa e rápida;
- (vi) “diagnosticabilidade”, relacionada à habilidade do sistema verificar seu estado atual para detectar falhas, diagnosticar suas causas e tomar decisões rápidas para não interromper a produção.

Existem também outras propostas que combinam os objetivos citados, tais como a personalização em massa que considera a flexibilidade de oferecer uma vasta quantidade de famílias de produtos modificados de acordo com as necessidades do cliente (TSENG;

HU, 2014). Segundo alguns pesquisadores (KOLBERG; ZÜHLKE, 2015; LEITÃO et al., 2015; MAŘÍK et al., 2015), na quarta revolução industrial, chamada de “Indústria 4.0”, novas tecnologias devem ser empregadas para integrar máquinas e humanos em cadeias de valor compondo uma rede de entidades (plantas industriais) e que devem fornecer serviços e produtos de forma autônoma (KOLBERG; ZÜHLKE, 2015). A Indústria 4.0 combina os objetivos dos paradigmas citados anteriormente, considerando a reconfiguração do sistema, a integração dos avanços tecnológicos relacionados a sistemas cibernéticos (LEITÃO et al., 2015) e a aplicação de conceitos relacionados à sistemas distribuídos.

Por outro lado, observa-se uma crescente tendência para sistemas produtivos distribuídos de forma dispersa, ou seja em localizações geográficas diferentes (JUNQUEIRA; VILLANI; MIYAGI, 2005; MELO et al., 2008; FATTORI et al., 2011). Um sistema produtivo disperso é composto por vários subsistemas que fornecem serviços específicos com certo grau de autonomia e conectados à internet. A Figura 2 apresenta um exemplo de sistema produtivo disperso, onde cada subsistema encapsula um serviço de manufatura: inspeção, alimentação (armazenagem e fornecimento de peças), montagem e transporte. Operadores podem atuar neste sistema produtivo disperso e clientes podem encomendar e acompanhar pedidos a distância. Assim, essa organização oferece serviços através de computação nas nuvens (*service cloud computing*) (NAGORNY; COLOMBO; SCHMIDTMANN, 2012; LI et al., 2010), isto é, por meio de *hardware* e *software* interligados por rede de acesso remoto, e onde atividades operacionais são descritas como serviços.

Figura 2 - Exemplo de sistema produtivo disperso

Adaptado de Fattori et al. (2011)b.

A consideração de novos paradigmas de sistema de manufatura em conjunto com a tendência

de operação de sistemas dispersos, levanta a questão de qual seria a arquitetura de controle que melhor atende às características requeridas a um controle de sistema reconfigurável de manufatura. Pois, as especificações da arquitetura servem de base para as atividades de desenvolvimento do controle que permita a reconfiguração de sistemas de manufatura que tende a ser disperso e que considere a integração/interação de sistemas cibernéticos.

A definição de uma arquitetura de controle consiste em especificar os componentes, suas interfaces, inter-relações e restrições. Dilts, Boyd e Whorms (1991) relacionam tipos de arquitetura de controle (Tabela 2), que são relacionadas às diferentes formas de organização de entidades e suas funções.

Tabela 2 - Tipos de arquitetura de controle

arquitetura de controle	características
<p>centralizada</p>	<p>utiliza um único componente central para decisões; resulta em “bom” gerenciamento e otimização do sistema; “baixa” velocidade de resposta; “difícil” utilização em sistemas complexos; “baixa” capacidade de expansão.</p>
<p>hierárquica</p>	<p>arquitetura normalmente distribuída em três níveis de controle; “boa” robustez; capacidade de predição “melhor” que na arquitetura centralizada; “fraca” resposta a falhas;</p>
<p>distribuída ou heterárquica</p>	<p>“rápida” resposta a falhas de solução local; “boa” velocidade de resposta; visão parcial do sistema; “fraca” resposta a falhas que exigem soluções que consideram o sistema como um todo.</p>
<p>hierárquica parcial dinâmica</p>	<p>“melhor” resposta a distúrbio/falhas; inclui interação entre o mesmo nível de controle; as entidades são fracamente acopladas, entretanto, ao menos alguma rotina deve ser fortemente acoplada.</p>
<p>legenda</p> <div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> </div> <div> <p>componentes do sistema de controle</p> <p>dispositivo sob controle</p> </div> <div style="margin-left: 20px; text-align: right;"> <p>(os termos entre aspas indicam uma qualificação relativa)</p> </div> </div>	

Adaptado de Dilts, Boyd e Whorms (1991).

Em geral, o controle de um sistema de manufatura convencional é organizado em estrutura hierárquica, conforme ilustrado na Figura 3 (ANSI/ISA, 1995). Nesta estrutura, redes de comunicação, tais como LAN e *fieldbus*, fornecem a malha de comunicação para troca de sinais entre os dispositivos. As funções de controle são divididas nas seguintes camadas:

- (i) nível corporativo (nível 4) – envolve o gerenciamento de informações, planejamento estratégico, determinação de níveis de estoque e outras funções relacionadas ao gerenciamento de alto nível da empresa. Ou seja, corresponde às funções relacionadas com as atividades do negócio necessárias para manter sustentável o sistema de manufatura. O nível 4 opera tipicamente dentro da escala de tempo de mês, semanas e dias;
- (ii) nível de fábrica ou produção (nível 3) – envolve agendamento, rastreamento de material em processo, roteamento de peças pelas máquinas, e utilização das máquinas. Ou seja, corresponde às funções relacionadas com as atividades que realizam a progressão dos processos para gerar os produtos finais. Inclui as atividades de manutenção de registros e coordenação dos processos. O nível 3 opera tipicamente dentro da escala de tempo de dias, turnos, horas, minutos e segundos, sobre áreas de centros de trabalho;
- (iii) nível de controle supervisorio (nível 2) – envolve controle e coordenação de grupos de máquinas e equipamentos de suporte trabalhando de modo sincronizado. Ou seja, corresponde às funções relacionadas com as atividades de monitoração e controle local dos processos físicos. O nível 2 opera tipicamente dentro da escala de tempo da ordem de horas, minutos, segundos ou suas divisões;
- (iv) nível de máquina (nível 1) – envolve o controle de máquinas de produção e estações de trabalho para a produção discreta de itens. Ou seja, corresponde às funções relacionadas com as atividades envolvidas na coleta de sinais e manipulação de itens físicos. O nível 1 opera tipicamente dentro da escala de tempo de segundos ou menos; e
- (v) nível de dispositivo (nível 0) – envolve as funções relacionados ao processo físico real.

Observa-se que a arquitetura hierárquica parcial dinâmica considera a execução de tarefas com previsibilidade dos resultados, tolerância e agilidade na resposta a falhas e aprendizagem. Segundo Giret e Trentesaux (2015), uma arquitetura de controle pode ser considerada hierárquica parcial dinâmica se a maioria das entidades são fracamente acopladas, entretanto, ao menos alguma rotina deve ser fortemente acoplada. De acordo com Brennan e Norrie (1999), nesta arquitetura parcial dinâmica, o mesmo planejamento centralizado e agendamento típico da arquitetura de controle hierárquica é usado, mas não é seguido de forma rígida, isto é, entidades de controle são capazes de cooperar para reagir a perturbações que independem dos planos globais. Em outras palavras, uma arquitetura é considerada parcial dinâmica se ela tem os aspectos de uma arquitetura heterárquica e considera um nível de controle supervisorio em certas situações. Um exemplo da arquitetura hierárquica são as arquiteturas baseadas no paradigma *Computer-Integrated Manufacturing* (CIM). Exemplo da arquitetura hierárquica parcial dinâmica pode ser observado em Borangiu et al. (2009) e Leitão e Restivo (2006).

Figura 3 - Arquitetura de controle de sistemas de manufatura.

Adaptado de ANSI/ISA (1995)

Uma arquitetura heterárquica pode ser observada no sistema holônico proposto por Ounnar et al. (2009).

Sob um ponto de vista diferente sobre a estrutura de controle, segundo Miyagi (1996), para desenvolver o sistema de controle, deve-se definir uma abordagem que permita especificar um sistema que atua (*at*) sobre o objeto de controle em função (*f*) de sinais de comando (*co*), detecção de sinais do objeto de controle (*de*) e do estado interno do sistema (*ei*), isto é, $at = f(co, de, ei)$. De acordo com Groover (2011), a abordagem deve considerar o sistema de controle definido como a regulação automática dos processos produtivos e de seus equipamentos associados, bem como a integração e a coordenação desses processos. Em relação ao tipo de controle envolvido (em lotes, contínuo ou discreto), nos níveis superiores (3 e 4) da arquitetura ANSI/ISA (1995) as funções de controle são semelhantes, pois mesmo admitindo o fato de que os produtos e processos sejam diferentes, o foco está na execução de um plano de tarefas pré-programadas. Entretanto, existem diferenças significativas destas funções nos níveis inferiores e intermediários (0 a 2). Groover (2011) utiliza o termo “variáveis” para denominar as variáveis de saída de um processo produtivo e “parâmetros” para denominar as variáveis de entrada. Desta forma, evidencia-se que a abordagem de Miyagi (1996) independe do nível relativo à estrutura de controle da ANSI/ISA (1995) e que junto com os trabalhos de Ramadge e Wonham (1989) confirmam que os sistemas de manufatura podem ser classificados como um sistema a eventos discretos, que é composto por vários subsistemas que executam diferentes processos produtivos que, por sua vez, interagem de diferentes formas. Um

subsistema de um sistema de manufatura consiste de vários componentes tais como máquinas de produção, sistema de manipulação de material, sistemas computacionais, e operadores humanos (HSIEH; LIN, 2013).

Conforme citado anteriormente, a reconfiguração de um sistema de manufatura envolve mudanças de recursos, *layout*, produtos, demandas e, além disso, mudanças devido à ocorrência de falhas. De fato, falhas são consideradas inevitáveis, pois componentes físicos envelhecem e sofrem com interferências externas nem sempre previsíveis, sejam ambientais ou humanas. Neste contexto, os projetos de *software* e *hardware* dos sistemas de controle são vítimas da fragilidade humana em trabalhar com grande volume de detalhes ou com deficiências de especificação. Considerando estes aspectos em relação a falhas, os pesquisadores têm estudado sistemas de controle de manufatura tolerantes a falhas.

Avizienis et al. (2001) definem falha (*fault*) como a causa física ou algorítmica do erro. O erro (*error*) é definido como a causa de um defeito (*failure*) o qual é definido como um desvio da especificação. Em sistemas produtivos, defeitos não podem ser tolerados, uma falha não necessariamente leva a um erro, o qual não necessariamente conduz a um defeito. A presença prévia de uma vulnerabilidade, isto é, uma falha interna que permita que um evento externo prejudique o sistema, é a condição necessária para que se tenha um erro e possivelmente um defeito subsequente. Esta relação entre falha-erro-defeito é representada na Figura 4.

Figura 4 - Relação entre falha-erro-defeito.

Adaptado de Avizienis et al. (2001).

Zhang e Jiang (2008) propõem uma forma de conceber um sistema de controle tolerante a falhas para assegurar a estabilidade e desempenho satisfatório do sistema de manufatura, não só quando todos os componentes estão funcionando normalmente, mas também em casos quando existem falhas. Ou seja, uma das decisões a considerar é se o sistema deve recuperar completamente o desempenho original ou pode aceitar algum grau de degradação do

desempenho após a ocorrência de uma falha. Os mecanismos de tolerância a falhas envolvem a detecção da falha, o estudo dos seus efeitos, identificação das causas e, finalmente, a reconfiguração do sistema.

Leitão e Restivo (2006) apresentam o problema de falha devido a indisponibilidade de atuadores que conduz a necessidade de reprogramação do processo produtivo no nível de planejamento e envolve a consideração de aspectos relacionados a reconfiguração. O objetivo neste caso é reparar o atuador e, em paralelo, encontrar soluções alternativas para reduzir o desvio do plano inicial. O estado do serviço e do atuador determina o tipo de reação à falha que deve ser acionado pelo sistema de controle. A reconfiguração envolve a reorganização dos recursos disponíveis na planta, com a possível adição ou a remoção de um recurso. Neste caso, a tarefa alocada ao recurso indisponível deve ser realocada a outros recursos disponíveis. Para a fase de decisão adota-se regras de inferência, tipo “se...então...” baseadas em raciocínio (KUIPERS, 1994).

A reconfiguração do sistema pode assim ser feita realocando funções e escolhendo caminhos alternativos de interação entre os processos. Em caso de ocorrência de falhas, a estratégia de reconfiguração é recuperar as funcionalidades do sistema para manter operações fundamentais. Em alguns casos, a ação de reconfiguração pode ser de tal forma que partes do sistema são desativadas sem afetar outras partes, levando o sistema para um estado seguro (conhecida como “degeneração” (ARAKAKI; MIYAGI, 2006)). Conforme ilustra a Figura 5, a atuação com degeneração (atd) é função (fd) do comando com degeneração (cod), da detecção com degeneração (ded) e do estado interno do sistema com degeneração (eid_i), isto é, $atd = fd(cod, ded, eid_i)$. O próximo estado interno com degeneração (eid_{i+1}) é função (g) da reconfiguração (rec) e do estado interno: $eid_{i+1} = g(rec, eid_i)$ e a reconfiguração para a degeneração (rec) é função (h) dos pontos críticos⁵ (pc) e da supervisão destes pontos críticos ($sucd$), isto é, $rec = h(sucd, pc)$.

Para permitir o diagnóstico de falhas, Sampath et al. (1996) apresentam um mecanismo chamado “diagnosticador” baseado em modelos de autômatos finitos para diagnóstico de falhas ou defeitos em SEDs. São construídos dois modelos: modelo geral de funcionamento do sistema e modelo do diagnosticador. É proposto um método para obter o modelo do sistema completo através da representação de componentes individuais e da informação fornecida por sensores. Para o modelo do “diagnosticador” são considerados “eventos observáveis” (aqueles que são relacionados a comandos de estratégias de controle) e “eventos não observáveis” (aqueles que podem estar relacionados a falhas) (DA SILVA; MIYAGI; FILHO, 2011).

Outro método de modelagem de mecanismos de tratamento de falhas é apresentado em

⁵causas de falhas que afetam a função fundamental do sistema

Figura 5 - Diagrama conceitual de sistema de controle com reconfiguração.

Fonte: (ARAKAKI; MIYAGI, 2006)

Miyagi e Riascos (2006). O método divide a abordagem destes sistemas em 3 módulos: módulo dos processos normais, módulo para detecção de falhas e módulo para tratamento de falhas. Têm-se basicamente quatro formas de executar o tratamento de falhas no sistema através da descrição do processo produtivo original (em situação normal) para incluir um sub-processo referente a detecção e o tratamento da falha: entrada condicionada, rota alternativa, recuperação inversa e recuperação direta. Modelos de processos baseados nos sinais emitidos pelos sensores no objeto de controle são desenvolvidos, tornando possível determinar uma relação entre as falhas e seus efeitos para a detecção dessas falhas, representando o conhecimento de um especialista.

Após definir a arquitetura, a classificação do tipo de controle e os mecanismos que permitem a reconfiguração em um sistema de manufatura; métodos de projeto que garantam a especificação desejada, devem ser adotados baseado em requisitos que definem recursos envolvidos no controle de processos e no controle de informações dos sistemas de manufatura. Bi et al. (2008) apresentam um levantamento desses requisitos:

- (i) composição por módulos de controle distribuídos e geograficamente dispersos;
- (ii) colaboração entre componentes para completar o objetivo do sistema como um todo;
- (iii) desenvolvimento para operar com sistemas heterogêneos, tais como sistemas operacionais, redes de comunicação, base de dados e protocolos;

- (iv) execução da adição, remoção, atualização de componentes (*hardware* ou *software*), quando a funcionalidade, capacidade e tecnologias são modificadas;
- (v) mudança de configuração de forma eficaz, rápida e automática, tendo consciência de alterações ocorridas no ambiente.

2.2 REDE DE PETRI

Modelos formais são elaborados para descrever e analisar o comportamento do sistema de manufatura e assegurar a verificação do atendimento dos requisitos especificados. Se comparadas com outras técnicas de descrição de Sistemas a Eventos Discretos (SEDs) como: cadeias de Markov, teoria das filas, álgebra de processo, álgebra max-plus, lógica temporal, teoria de linguagem e/ou autômatos; a Rede de Petri (RdP) possui uma capacidade equivalente de modelagem. Entretanto, o diferencial é sua facilidade de visualização do sistema e a possibilidade de conversão para programas em linguagens de controladores industriais. Além disso, ela é uma ferramenta de comunicação entre equipes relacionadas ao projeto (ZURAWSKI; ZHOU, 1994; REISIG, 1985; MURATA, 1989).

A RdP tem sido usada na modelagem e análise de diferentes tipos de sistemas e aplicações tais como: protocolos distribuídos (KANESHIRO et al., 2008), processos industriais (ZURAWSKI; ZHOU, 1994; NASSAR et al., 2008), tratamento de falhas (MORALES; MELO; MIYAGI, 2007; LIRA et al., 2008; PORCARELLI et al., 2004; MIYAGI; RIASCOS, 2006; SQUILLANTE et al., 2012), fluxo de processos (KIEPUSZEWSKI; HOFSTED; AALST, 2003), controle supervisório (LEE; ZHOU; HSU, 2005; MELO; MORALES; MIYAGI, 2008), sistemas produtivos dispersos (FATTORI et al., 2011; DA SILVA et al., 2014), e sistemas multiagentes para manufatura (LEITÃO; RESTIVO, 2006; HSIEH, 2009).

O Production Flow Schema (HASEGAWA et al., 1999) é uma extensão de RdP que visa a descrição estrutural e hierárquica de um sistema em um nível relativamente alto de abstração e é uma técnica efetiva para auxiliar a construção de um outro modelo em RdP para descrever o comportamento dinâmico e detalhado do sistema. Esta abordagem pode ser explorada com a extensão de RdP proposta por (GOMES et al., 2007) chamada de Input Output Place Transition, a qual possui um *software* disponível na internet com interface relativamente intuitiva para edição e análise por simulação do comportamento dinâmico de um sistema. Isto permite o envolvimento e a colaboração de diferentes desenvolvedores de acordo com a visão de um ambiente de *cloud manufacturing* (LI et al., 2010; DA SILVA et al., 2015). Além disso, a técnica tem a vantagem de permitir transformar os modelos IOPT para programas executáveis em controladores industriais.

2.2.1 Production Flow Schema

O *Production Flow Schema* (MIYAGI, 1996) é uma extensão de rede de Petri chamada de canal-agência (REISIG, 1985) devidamente interpretada para sistemas produtivos, tais como os sistemas de manufatura. Constitui uma técnica para desenvolver uma descrição estrutural hierárquica de um sistema a partir de uma visão conceitual em um nível relativamente alto de abstração sem o detalhamento do seu comportamento dinâmico. Ou seja, o PFS é o primeiro passo para se aplicar a técnica de refinamentos sucessivos através de uma abordagem *top-down*, de maneira que seja possível inserir progressivamente no modelo o detalhamento dos processos do sistema. O objetivo é representar o fluxo de operações tendo como referência a evolução de um determinado processo. A partir do PFS pode ser gerado um outro modelo em RdP para descrever o comportamento dinâmico do sistema (JUNQUEIRA; MIYAGI, 2006). A ideia é que uma RdP gerada a partir do PFS apresente propriedades estruturais que atendam especificações próprias de sistemas produtivos reais. Os elementos do PFS, ilustrados na Figura 6, são os seguintes:

- (i) elementos ativos chamados *atividades*⁶ : representam um subprocesso ou um componente ativo do sistema capaz de realizar transformações. As *atividades* podem agregar uma ou mais operações e pode ser refinada em *subatividades*. Correspondem a um macro elemento delimitado por “[” e “]”.
- (ii) elementos passivos chamados *distribuidores*: representam componentes passivos, capazes de armazenar, permanecer em determinados estados ou tornar os itens visíveis. É indicado graficamente por uma circunferência. Entre duas *atividades* sempre existe um elemento *distribuidor*.
- (iii) *arcos* que conectam *atividades* e *distribuidores* representando o fluxo de materiais e/ou informações, explicitam uma relação lógica entre estes elementos. No PFS se tem os fluxos primários, que estão conectados diretamente aos símbolos “[” e “]”, e os fluxos secundários, que estão conectados à parte interna da *atividade* (a representação deste fluxo é opcional).

De acordo com Hasegawa et al. (1999), o PFS associado com uma RdP Lugar/Transição (RdP L/T) (descrita na próxima subseção) permite modelar o fluxo bem como comportamento de um sistema de manufatura em uma forma mais simplificada e transparente do que o uso de metodologias baseadas em uma única representação de RdP.

⁶Termos relacionados a rede de Petri estão destacados em fonte Courier.

Figura 6 - Elementos do PFS.

Portanto, o PFS é uma técnica para especificação de sistema produtivo em nível conceitual e ela é efetiva como base para a derivação de modelos em RdP com “boas propriedades”: limitabilidade, reiniciabilidade e vivacidade (CARDOSO; VALETTE, 1997). Observa-se também que o PFS é adequado para definir uma sistemática para projeto e comunicação entre envolvidos no projeto do controle de um sistema reconfigurável de manufatura.

2.2.2 Rede de Petri Lugar/Transição (RdP L/T)

A RdP L/T é um grafo bipartido composto de lugares (representados por círculos), transições (representados por barras), arcos orientados (interligando os elementos anteriores) e marcas (utilizadas para definir o estado da RdP).

O peso de um arco indica o número de arcos orientados de mesmo sentido entre dois elementos.

Considerando o sentido dos arcos, chama-se lugares de entrada de uma transição, os lugares que possuem arcos no sentido destes lugares para a citada transição e, chama-se lugares de saída de uma transição, os lugares que possuem arcos no sentido da citada transição para estes lugares.

A dinâmica de um sistema modelado em RdP é descrita pela execução de uma rede de Petri é caracterizada pela variação do número de marcas nos lugares. A presença ou ausência de marcas nos lugares determinam o disparo das transições. A evolução dinâmica ocorre através do disparo de transições que alteram a marcação da RdP. Quando uma transição dispara os lugares de entrada e de saída da transição têm a sua marcação (número de marcas) reduzida e aumentada, respectivamente, pelo peso dos arcos que conectam os lugares às transições (Figura 7).

As regras básicas para a modelagem da evolução dos estados da RdP incluem:

- (i) uma transição só pode ser disparada se seus lugares de entrada estão marcados (eles devem ter pelo um número de marcas igual ou maior que o peso dos respectivos arcos) e seus lugares de saída têm capacidade suficiente para receber

Figura 7 - Evolução dinâmica derivada do disparo de uma transição

novas marcas em número equivalente ao peso dos respectivos arcos;

- (ii) no disparo, a transição remove um número de marcas de cada lugar de entrada igual ao respectivo peso do arco (de entrada da transição) e coloca um número de marcas em cada lugar de saída igual ao respectivo peso do arco (de saída da transição); e
- (iii) as marcas não têm identificação e são usadas apenas para indicar se a condição representada é verdadeira (marcada) ou falsa (não marcada).

A Figura (8) ilustra os principais elementos da RdP L/T apresentada nesta seção.

Figura 8 - Elementos de uma RdP L/T.

A Rede de Petri L/T é uma 5-tupla (P, T, F, W, M_0) (MURATA, 1989; ZURAWSKI; ZHOU, 1994; ADAM; ATLURI; HUANG, 1998), com $P = p_1, p_2, p_3, \dots, p_r$ ($r \in \mathbb{N}^*$) e $T = t_1, t_2, t_3, \dots, t_s$ ($s \in \mathbb{N}^*$) como conjuntos finitos, não nulos e disjuntos. P é o conjunto de lugares e T é o conjunto de transições. $F = P \times T \cup T \times P$ é o conjunto dos relacionamentos entre lugares e transições e é também chamado de conjunto de arcos orientados. $W \rightarrow \mathbb{N}^*$ é um conjunto de números naturais não nulos que representam os pesos de cada arco orientado. A ausência de indicação é considerada como peso unitário. M_0 é um vetor de números naturais que representa a quantidade inicial de marcas nos lugares, que também é conhecido como marcação inicial (MURATA, 1989; LI; ZHOU, 2008; ZURAWSKI; ZHOU, 1994; YOO; JEONG; CHO, 2010; HAN et al., 2008; XU et al., 2007), isto é, M é a marcação da rede, onde $M = m_{p_1}, m_{p_2}, \dots, m_{p_r}$ ($r \in \mathbb{N}^*$) é definida pelo vetor das marcações de cada lugar p_i contendo o número de marcas e M_0 é a marcação inicial.

A RdP possui propriedades que quando interpretadas no contexto do sistema modelado, permitem ao desenvolvedor do sistema identificar a presença ou ausência de comportamentos e funcionalidades específicas do domínio da aplicação do sistema em projeto. As propriedades podem ser distinguidas em comportamentais e estruturais (ZURAWSKI; ZHOU, 1994).

As propriedades comportamentais são aquelas que dependem do estado inicial da RdP, ou seja, do conjunto de marcas iniciais (marcação M_0). Já as propriedades estruturais não dependem do conjunto de marcas iniciais da RdP, mas sim de sua topologia (ZURAWSKI; ZHOU, 1994).

A estrutura de uma RdP L/T é uma 4-tupla (P, T, F, W) (MURATA, 1989). Uma RdP pode ser dita “ordinária” se $\forall f \in F, W(f) = 1$ e, por isso, sua estrutura é denotada como $N = (P, T, F)$ (LI; ZHOU, 2008). A RdP (P, T, F, W) é dita “pura” se, e somente se, não há *loops*, isto é, $\nexists (x, y) \in (P \times T) \cup (T \times P), (x, y) \in F \wedge (y, x) \in F$ (LI; ZHOU, 2008).

Outra representação formal da RdP é por meio de matriz de incidência e equações de estados. A matriz de incidência em uma RdP (P, T, F, W) é representada por $[c_{ij}]$ indexada por P e T tal que $p_i \in P, t_j \in T$ e $f_{ij} \subseteq (P \times T) \cup (T \times P)$ com f_{ij} o arco orientado ligando o lugar p_i à transição t_j ou a transição t_j ao lugar p_i , $c_{ij} = W(f_{ij})$ (MURATA, 1989).

Em um grafo (P, T, F, W) o pré-conjunto $x \in P \cup T$ é definido como $\bullet x = \{y \in P \cup T | (y, x) \in F\}$. O pós-conjunto $x \in P \cup T$ é definido como $x^\bullet = \{y \in P \cup T | (x, y) \in F\}$. Os pré-conjuntos ou pós-conjuntos de um conjunto de elementos são definidos como a união dos pré-conjuntos ou pós-conjuntos desses elementos, respectivamente (LI; ZHOU, 2004).

No caso de uma RdP ordinária e pura, a matriz $[c_{ij}]$ é tal que $c_{ij} = 1$ se $t_j \in \bullet p_i \setminus p_i^\bullet$, $c_{ij} = -1$ se $t_j \in p_i^\bullet \setminus \bullet p_i$ e $c = 0$ nos demais casos (LI; ZHOU, 2008; ZURAWSKI; ZHOU, 1994).

A dinâmica de uma RdP é caracterizada pela mudança de estados da mesma e a simplicidade das regras de mudança de estados da RdP é a base do sucesso dessa ferramenta na modelagem e análise de sistemas dinâmicos (MURATA, 1989).

Em uma RdP ordinária e pura, uma transição $t \in T$ está habilitada em M (simbolicamente $M[t >)$ se, e somente se, $\forall p \in \bullet t : M(p) > 0$ e $\forall p \in t^\bullet : M(p) = 0$. Uma transição $t \in T$ habilitada pode ser disparada e como resultado desse disparo tem-se uma nova marcação M' , que é denotado por $M[t > M'$, com $m'_p = m_p - 1$ se $p \in \bullet t \setminus t^\bullet$; $m'_p = m_p + 1$ se $p \in t^\bullet \setminus \bullet t$; e $m_p = m'_p$, nos demais casos, para todo $p \in P$ (LI; ZHOU, 2004; ADAM; ATLURI; HUANG, 1998).

(N, M) é chamado de rede marcada e limitada se, e somente se, $\exists k \in \mathbb{N}^+, \forall M \in R(N, M_0)$ e $\forall p \in P$, tal que $m_p \leq k$, em que $R(N, M_0)$ é o conjunto das marcações na rede N alcançáveis a partir de M_0 (LI; ZHOU, 2008).

2.2.3 Input Output Place Transition

A IOPT é baseada em sinais/eventos de entrada e saída (*input and output signals/events*) que permitem a descrição de lógicas de dispositivos de controle; suporta execução determinística autônoma que facilita a modelagem de programas de controladores industriais; e, se houver conflitos, IOPT permite o uso de prioridades associadas a transições ou de combinações lógicas específicas (RIBEIRO et al., 2011; PAIS; BARROS; GOMES, 2014).

De fato, existem outras extensões da RdP como o MFG, GRAFCET e SFC (MIYAGI, 1996) concebidos e já adotados na prática (ASATO et al., 2015) em muitos países com os mesmos fins da IOPT. Entretanto, uma vantagem adicional é que o modelo em IOPT pode ser automaticamente convertido para programas de controladores industriais (GOMES et al., 2007) utilizando um *software* aberto e com documentação relativamente acessível e disponível na internet. Especificamente, a IOPT *web tool* permite gerar programas em códigos C, VHDL or JavaScripts. O gerador de código C produz um arquivo comprimido que segue a sintaxe ANSI C e contém os seguintes sub-arquivos:

- (i) *net_types.h*, que é a definição do tipo de dados e declarações de funções, contendo *places marking, input and output signals e events*;
- (ii) *net_main.c*, que é função principal do ciclo de execução;
- (iii) *net_io.c*, que é o código *input and output* usado para ler os valores de *input signals* e escrever os *output signals* do *hardware*;
- (iv) *net_functions.c*, que tem as funções para implementar todas regras semânticas da IOPT: *transition firing, guards, events, output actions, etc.*;
- (v) *net_exec_step.c* que tem a função que executa um passo da IOPT, usando as várias funções definidas em *net_functions.c*;
- (vi) *makefile* que é um arquivo opcional Unix com instruções para construir o projeto.

Todos os códigos produzidos podem ser compilados na maioria dos sistemas operacionais, sem qualquer mudança, exceto o arquivo *net_io.c* que deve ser adaptado ao domínio de aplicação.

IOPT estende a classe de RdP L/T e é uma tupla $(P, T, F, W, M_0, TA, WTest, priority, isg, ie, oe, osc)$ onde:

- (i) a definição de P, T, F, W e M_0 é a mesma da RdP lugar/transição;
- (ii) TA é um conjunto de arcos teste, tal que $TA \subseteq (P \times T)$;

- (iii) *WTest* é uma função peso associada aos arcos testes, $WTest : TA \rightarrow \mathbb{N}^*$;
- (iv) *priority* é uma função parcial denominada prioridade que associa inteiros não-negativos a transições, $priority : T \rightarrow \mathbb{N}^*$;
- (v) *isg* é uma função input signal guard que associa funções booleanas (de um conjunto finito BE) compostas por input signals (de um conjunto finito IS) a transições, $isg : T \rightarrow BE$, onde $\forall BE \in isg(T), Var(BE) \subseteq IS$;
- (vi) *ie* é uma função composta por input events de um conjunto finito (IE) associado a transições, $ie : T \rightarrow IE$;
- (vii) *oe* é uma função de output events de um conjunto finito (OE) associado a transições, $oe : T \rightarrow OE$;
- (viii) *osc* é uma função das condições de output signals de lugares dentro de um conjunto de regras (*RULES*), $osc : P \rightarrow P(RULES)$, onde $RULES \subseteq (BES \times OS \times \mathbb{N}^*)$, $BES \subseteq BE \ \forall e \in BES, Var(e) \subseteq ML$ sendo que: *OS* é o conjunto finito de *output signals* (*os*), $Var(e)$ é a função que retorna o conjunto de variáveis em uma dada expressão *e*, e *ML* é o conjunto de identificadores para cada marcação de lugar após uma dada execução de um passo. Portanto, cada marcação de lugar tem associado um identificador, o qual é usado quando for executar o código gerado.

Em síntese, as principais diferenças entre a RdP L/T e IOPT são os seguintes elementos:

- (i) *guard*, que representa uma expressão lógica que permite ou não o disparo da transição;
- (ii) *priority*, uma explicita a especificação que fornece uma solução para possíveis conflitos;
- (iii) conjunto de *input events*, que é uma lista de eventos que devem ocorrer para uma transição disparar se seu *guard* é atendido;
- (iv) conjunto de *output events*, que é uma lista de eventos que devem ocorrer pelo disparo de uma transição; e
- (v) IO, uma lista de *input* e *output signals*.

O uso de aplicativos computacionais voltados para a análise via simulação de RdP permitem também que se construa um modelo do processo com tempo relativamente reduzido, com vantagens na visualização. Além de se obter estatísticas e verificar o atendimentos das propriedades da RdP, é possível visualizar o a dinâmica funcional dos modelos e

detectar possíveis erros ou problemas no sistema, facilitando o entendimento dos *stakeholders* (envolvidos) no projeto. A IOPT *web tool* (GOMES et al., 2007) tem uma interface gráfica intuitiva e relativamente fácil para edição e análise via simulação dinâmica de um modelo IOPT, e verificação das propriedades, além de ser um software aberto escrito em linguagem Java.

Figura 9 mostra a tela de trabalho da IOPT *web tool*. Este programa possibilita executar a simulação passo-a-passo ou continuamente. O programa grava o histórico de todos os passos executados em um arquivo que pode ser salvo. Este histórico também permite o usuário avançar ou retornar passos na simulação. A tela de trabalho é dividida em: um *toolbox* na esquerda, abas com o estado dos elementos IOPT na direita e o modelo gráfico ao centro. Em tempo de execução, os elementos do modelo mudam a sua cor de acordo com a habilitação dos elementos envolvidos, conforme lista a Tabela 3 para lugares e transições. *Input* e *Output Events* podem ser acionados tanto no modelo gráfico central como nas abas localizadas a direita, os valores dos *output events* são automaticamente restabelecidos após cada execução, ao contrário de *input signals* que mantém os valores anteriores.

Figura 9 - Tela de trabalho do *web tool* IOPT.

Além de permitir a simulação/edição de modelos IOPT e a obtenção de programas para controladores industriais, a *webtool* considera a representação de sistemas que possuem processos caracterizados como *Globally-Asynchronous Locally-Synchronous* (GALS) (sistemas GALS) (MOUTINHO; GOMES, 2012). Para representar sistemas GALS na IOPT *web tool*,

Tabela 3 - Esquema de cores do *software* de edição/simulação de IOPT.

elemento	cor (edição)	cor (simulação)
lugar	amarelo (marcas são indicadas por números)	0 marcas: amarelo 1 marca: vermelho >1 marca: violeta
transição	ciano	I/O atendidos: amarelo habilitada: laranja I/O atendidos e habilitada: verde qualquer outro caso: ciano

deve-se definir quais modelos são localmente síncronos através da indicação de um *time domain* (td) para cada `lugar` ou `transição`. Sistemas GALS são representados na IOPT considerando processos locais síncronos que se comunicam de forma assíncrona com outros processos globais e distribuídos através de um elemento chamado *Asynchronous Channel* (AC). ACs são representados por `lugares` em formato de “nuvem” (vide centro na Figura 9) e são conectados às `transições` por meio de arcos tracejados, para diferenciar de outros tipos de arcos e `lugares`. A IOPT *web tool* permite decompor sistemas GALS sincronizados por ACs através de uma aba chamada “decomposes GALS”, a qual decompõe o modelo global em sub-modelos locais que passam a ser sincronizados por `input` e `output events`.

2.3 HOLONS E AGENTES

Uma das formas de facilitar o projeto de sistemas de controle é considerar o reuso de modelos, sua interoperabilidade e portabilidade, especificando uma automação baseada em componentes pré-definidos (TOMMILA et al., 2005). Busca-se evitar a repetição de tarefas com sobreposição de escopos na especificação dos domínios. Uma abordagem baseada em componentes permite a definição de uma linguagem comum e o reuso de modelos.

O *Holonic Manufacturing System Consortium* (CHRISTENSEN, 2003) trabalhou na aplicação dos conceitos propostos por Koestler (1968) para desenvolver uma nova geração de sistemas de manufatura e seu controle baseado em componentes. Koestler descreve um elemento básico na composição de sistemas como em organismos vivos e sociais, considerando que o conceito de partes e do todo existe no domínio da vida, e propôs a palavra “holon” para representar esta natureza híbrida. Ele identificou duas características importantes de um holon: autonomia, em que a estabilidade do holon resulta de sua habilidade de agir de forma autônoma e colaborativa; e cooperação, que é a combinação das habilidades os holons para atingir objetivos comuns e define esta colaboração como “holarquias”. Ou seja, de acordo com o ponto de vista do projetista, um holon pode formar holarquias que podem ser consideradas

um conjunto de holons que interagem para colaborar entre si ou ser decomposto em holons atômicos, os quais não podem ser mais decompostos (GIRET; BOTTI, 2009), conforme ilustra a Figura 10.

Figura 10 - Representação de holons e holarquias

Na Figura 11, tem-se um exemplo de aplicação do conceito de holarquias em sistemas de manufatura apresentado em Leitão e Restivo (2006), onde o holon supervisor “SH” e operacional “OH” podem ser decompostos em outros holons supervisores “SHs” e “OHs” e, desse modo, esses holons podem representar desde células de manufatura a dispositivos de controle.

Figura 11 - Recursividade em sistemas holônicos.

Adaptado de Leitão e Restivo (2006).

Da área de teoria de sistemas segundo Russell et al. (1995), um agente é tudo o que pode ser considerado com capacidade de percepção de seu ambiente por meio de sensores e execução de ações sobre este ambiente por intermédio de atuadores, conforme ilustrado na Figura 12. O termo percepção se refere às entradas perceptivas do agente em um dado momento e a escolha de uma ação, a ser realizada pelos atuadores do agente, pode depender da sequência de percepções ou função de agente (nesta figura esta função está representada por uma “?”).

Um agente precisa ser capaz de agir sem a intervenção direta de humanos ou outros agentes, e deve ter controle sobre suas próprias ações e estados internos (JENNINGS; WOOLDRIDGE, 1998; JENNINGS; BUSSMANN, 2003).

Figura 12 - Arquitetura básica de um agente.

Adaptado de Russell et al. (1995).

Outra definição é apresentada por Ferber (1999), que considera um agente uma entidade autônoma (física ou virtual), capaz de agir sobre seu ambiente e comunicar com outras entidades. Para agir, um agente processa uma representação parcial de seu ambiente ou é capaz de sentir parte do ambiente. Através de interações com outros agentes ou com o ambiente, os agentes são capazes de produzir comportamentos coletivos relativamente complexos. Portanto, sistemas multiagentes possuem vários agentes organizados de forma heterárquica que interagem para atingir o objetivo do sistema como um todo. A interação é feita através de definições de funções e regras para troca de informação, tomada de decisão e funções de controle.

Uma plataforma de *software*⁷ ou *framework*⁸ baseada em sistema multiagente (ou seja, uma plataforma de *software* ou *framework* formada por dois ou mais agentes) deve executar funções de controle de maneira autônoma e inteligente com capacidades de comunicação, cooperação, sincronização, reconfigurabilidade e flexibilidade. Um sistema multiagente pode ser composto por vários outros sistemas multiagentes e, neste caso, pode-se ter um sistema multiagente federado (FERBER, 1999; LEITÃO, 2009).

A arquitetura de controle baseada em holons e agentes, conhecida como *Holonic and Multi-Agent System* (ZHANG et al., 2000; COLOMBO; NEUBERT; SCHOOP, 2001), é uma solução que atende grande parte dos requisitos de projeto de um controle de SRM (MAŘÍK et

⁷Um conjunto de ferramentas de programação e programas computacionais que podem ser usados para alavancar o reuso e o rápido desenvolvimento de novos aplicativos, e que define o ambiente operacional, a arquitetura de todos os aplicativos desenvolvidos baseados nela, e um conjunto de regras de desenvolvimento para aperfeiçoá-la.

⁸Um conjunto de objetos que colaboram para realizar uma funcionalidade para um determinado domínio de um subsistema (JOHNSON; RUSSO, 1991).

al., 2015). Nesta arquitetura, um holon é uma abstração de um agente, e ambos, no nível de controle inferior das operações produtivas de máquinas podem ser vistos como estruturas de códigos executáveis adaptadas de acordo com a implementação. De fato, a maioria das técnicas utilizadas para implementação de sistemas holônicos estão presentes em sistemas multiagentes (JENNINGS; BUSSMANN, 2003), pois ambos se referem a entidades autônomas que têm de operar em um ambiente, reagindo às mudanças e objetivos, e que interagem uns com os outros formando holarquias (GIRET; BOTTI, 2004). No entanto, também existem algumas diferenças (LEITÃO, 2009):

- (i) sistemas multiagentes têm as suas raízes na ciência da computação enquanto sistemas holônicos no domínio das tecnologias de CIM;
- (ii) sistemas holônicos é uma técnica com abordagem essencialmente conceitual enquanto sistemas multiagentes é uma abordagem mais tecnológica, ou seja, um agente representa entidades de *software* e holon é uma abstração da forma de integração de recursos físicos;
- (iii) um holon pode formar holons maiores ou serem decompostos em outros holons de nível inferior, isto é, tem característica de recursividade, que não é considerado para um agente.

Entretanto, é possível tirar proveito das características comuns de sistemas holônicos e sistemas multiagentes e, por outro lado, conciliar as diferenças porque são características que se complementam. Além disso, plataformas de sistemas multiagentes são adequadas para implementação de sistemas holônicos. Segundo Colombo, Neubert e Schoop (2001), esta combinação pode ser efetiva se for focada sobre a funcionalidade do sistema em vez da abordagem do controle centralizado de uma automação convencional. A Figura 13 ilustra uma forma de como implementar esta combinação e cooperação entre os componentes.

2.3.1 Análise de sistemas multiagentes

Entre as técnicas de análise de sistemas de manufatura, a simulação utiliza modelos matemáticos/gráficos que descrevem um cenário e onde hipóteses podem ser testadas repetidamente para verificar os resultados. A simulação tem várias vantagens para o projeto de sistemas de manufatura como (LEITÃO et al., 2009):

- (i) o sistema, ou parte dele, pode ser testado, depurado e validado sem necessidade de usar o sistema físico real;
- (ii) a reprodução de diferentes cenários, testes de condições anormais ou perigosas, podem ser feitas de uma forma segura através de modelos virtuais;

Figura 13 - Representação dos níveis de controle em um HMAS.

Adaptado de Mařík e Lažanský (2007).

- (iii) os dados obtidos podem ser utilizados para treinamento de operadores e atividades de operação/manutenção;
- (iv) a simulação pode ser repetida várias vezes quanto for necessário para o entendimento e ajuste das soluções de controle;
- (v) o tempo de simulação do processo pode ser comprimido (isto é, acelerando a execução temporal do modelo) para permitir obter resultados que através do sistema físico seria bastante demorado.

Além disso, a visualização estrutural e funcional do sistema é outra vantagem da simulação, assim operadores e outros interessados podem entender melhor um sistema de manufatura, pois é possível acompanhar a dinâmica do sistema por recursos gráficos. A detecção de erros, falhas e resolução de mau funcionamento podem ser visualizados e solucionados antes da implementação. Ou seja, a simulação estende o processo de modelagem para análise comportamental/funcional do sistema de manufatura.

Existem várias plataformas para simulação de sistemas de manufatura, tais como Arena, Promodel, Matlab (KLINGSTAM; GULLANDER, 1999). Entretanto, visto que neste trabalho, o uso da simulação é para análise de sistemas de controle que são projetados baseado em agentes, observa-se que a escolha mais adequada é o uso de plataformas para tal fim; pois, a simulação de sistemas multiagentes requer o atendimento de requisitos, tais como arquitetura distribuída e interfaces para a análise da colaboração entre as entidades do sistema.

Vrba (2006) apresentam o *software Manufacturing Agent Simulation Tool* (MAST), focando na modelagem do roteamento dinâmico de processos em sistemas de manufatura. MAST foi

usado para simular uma célula de empacotamento e um sistema de transferência de *pallets*. Bal e Hashemipour (2009) apresentam outro exemplo utilizando uma abordagem baseada em realidade virtual para simular uma aplicação de sistemas holônicos em uma indústria de fundição. Estes exemplos apresentam certa complexidade na simulação, pois os *software* são desenvolvidos especificamente para cada caso, isto é, requerem significativo esforço para analisar uma solução de HMAS em outras aplicações.

Existem vários softwares de licença livre para simular sistemas multiagentes com diferentes funcionalidades, interfaces gráficas e diferentes linguagens de programação, tais como Repast (NORTH et al., 2007), Swarm (MINAR et al., 1996), NetLogo (TISUE; WILENSKY, 2004) e Mason (LUKE et al., 2005). Tobias e Hofmann (2004) apresentam um comparativo destes *software* de sistemas multiagentes de licença livre e que utilizam a linguagem Java. São comparados os tipos de licença, qualidade de documentação, tipo de suporte, manutenção e/ou viabilidade futura, interface de modelagem, controle da simulação e facilidade de uso. As ferramentas Repast e Swarm obtiveram a maior classificação.

Castle e Crooks (2006) apresentam outro comparativo entre plataformas de sistemas multiagentes onde as plataformas foram classificadas em grupos. No primeiro grupo foram incluídos *software* de código aberto: Mason, Repast e Swarm, os quais usam linguagem Java. No segundo grupo foram incluídos *software shareware/freeware*, tais como StarLogo e NetLogo. Os autores concluem que Starlogo e Netlogo são aqueles que apresentam maior facilidade de programação e acesso à documentação/suporte.

Barbosa e Leitão (2011) apresentam um HMAS que considera cooperação entre holons de produtos e recursos para atender um determinado plano de produção. Para validação, os autores fazem um estudo comparativo dos *software* disponíveis para análise de sistemas multiagentes e concluem que Netlogo é a melhor opção. A Tabela 4 apresenta um resumo das características dos *software* que foram considerados neste estudo.

Tabela 4 - Comparativo entre *software* para simulação de sistemas multiagentes.

	MASON	Netlogo	Swarm	Repast
maturidade	ruim	razoável	bom	razoável
esforço de programação	razoável	bom	razoável	ruim
mudança de propriedades	ruim	razoável	ruim	bom
interface do usuário	ruim	bom	ruim	bom
velocidade	bom	razoável	razoável	bom
documentação	bom	bom	razoável	razoável

Adaptado de Barbosa e Leitão (2011).

O NetLogo é executável em uma *Java virtual machine* e programado em linguagem Logo

(FEURZEIG; PAPERT; LAWLER, 2011), ou seja, Netlogo pode ser executado em plataformas de sistemas operacionais diferentes com facilidade de programação. Basicamente, Netlogo considera três tipos de agentes: estacionário (*patches*), móveis (*turtles*), e *link*, o qual permite unir os dois agentes anteriores formando redes, grafos e conjuntos agregados. Os *patches* são organizados em uma grade formando a região (*layout*) onde a dinâmica dos agentes *turtles* pode ser observada (TISUE; WILENSKY, 2004).

NetLogo pode ser usado para o estudo de diferentes sistemas e o usuário pode fazer várias alterações na representação do sistema. Por exemplo, pode ser alterado o tamanho, formato e cor dos agentes *patches* e *turtles*. A área de trabalho do NetLogo é composta de três abas: interface, informação e procedimentos. A aba interface inclui uma parte gráfica para permitir a inserção de botões, gráficos e a visualização do *layout* do sistema e o cenário. A aba informação é usada para incluir descrições sobre o cenário, tais como objetivo, funções, procedimentos e possíveis erros. A intenção é fornecer suporte para outros usuários entenderem o sistema modelado. A aba procedimentos é onde o código referente à especificação do sistema é incluído e os comandos são editados, definindo as características e o comportamento esperado. NetLogo utiliza *ticks* (unidade de tempo computacional) para simulação ao invés de tempo real, assim os resultados não são dependentes da velocidade do processador, permitindo comparar com relativa facilidade diferentes cenários de simulação. A Figura 14 apresenta a ilustração das três abas para o exemplo clássico do “jantar dos filósofos”, o qual está disponível na biblioteca do *software* (WILENSKY, 2003). Neste exemplo, os filósofos precisam compartilhar garfos para jantar e todos não podem realizar isto simultaneamente, o que demanda uma solução que deve considerar compartilhamento de recursos, concorrência e conflito.

2.3.2 Projeto e implementação de sistemas multiagentes

Para projeto de sistemas multiagentes pode-se utilizar o método de projeto orientado a objetos (BOOCH; RUMBAUGH; JACOBSON, 2005) com passos pré-definidos seguindo abordagens *bottom-up* ou *top-down* (ROBINSON, 1992) e as especificações da *Foundation for Intelligent Physical Agents* (FIPA) (FIPA, 2002). Estas especificações da FIPA envolvem protocolos para comunicação entre os agentes, tais como *querying*, *voting*, *negotiation*, e *auctions*, e o FIPA-CNP baseado no *Contract-net protocol* (CNP) proposto por Smith (1980), que é um protocolo do tipo *auction* caracterizado por *one-sided*, *first-price*, e *sealed-bid auction*. O desenvolvimento de sistemas multiagentes requer a implementação de funcionalidades, tais como transporte de mensagens, codificação e análise sintática, ontologias, e gestão do ciclo de vida dos agentes, o que aumenta o esforço de construção desses sistemas.

A definição da ontologia é fundamental para colaboração entre equipes/entidades dispersas,

Figura 14 - Abas do *software* NetLogo do exemplo jantar dos filósofos.

Fonte: (WILENSKY, 2003)

distribuídas e autônomas, tais como sistemas baseados em HMAS. Padrões de ontologia para sistema de manufatura são propostas por ANSI/ISA 88 (ANSI/ISA, 1995), OONEIDA (VYATKIN; CHRISTENSEN; LASTRA, 2005), MASON (LEMAIGNAN et al., 2006), ADACOR (LEITÃO; RESTIVO, 2006) e Obitko et al. (2010).

A maioria das interações em um HMAS é baseada no *Contract Net Protocol* (CNP), o

qual é baseado na noção de contratos em uma estrutura descentralizada onde os agentes podem assumir dois papéis: *manager* e *contractor*. O *manager* divide a tarefa de controle em subtarefas e envia mensagens para os *contractors*, as quais também tem a tarefa de apresentar as propostas para os *managers*. Após realizar inferências pré-definidas sobre as propostas, o *manager* contrata a parte que oferece a melhor proposta. O CNP possui as seguintes fases: (i) *announcement*, em que o *manager* ao ser requisitado deve decompor a tarefa requisitada em subtarefas e enviar anúncios aos *contractors* que possuam as habilidades para resolvê-las; (ii) *bidding*, em que os *contractors* enviam propostas ao *manager* em resposta ao *announcement*; e (iii) *awarding*, em que o *manager* valida as propostas recebidas e seleciona as mais adequadas estabelecendo acordos para execução das subtarefas.

Todas as citadas especificações/funções tornam relativamente complexa a construção de sistemas multiagentes e plataformas são desenvolvidas para este fim. Vrba (2003) faz uma comparação entre as plataformas e conclui que o *Java Agent DEvelopment framework* (JADE) (BELLIFEMINE; POGGI; RIMASSA, 1999) é a mais adequada, superando as demais e, especificamente, o principal concorrente, FIPA-OS (POSLAD; BUCKLE; HADINGHAM, 2000), quanto ao aspecto de velocidade de entrega de mensagens. JADE é uma plataforma livre com características para suportar o gerenciamento de sistemas multiagentes e a implementação de programação baseada em regras. A interconectividade é baseada no protocolo TCP/IP permitindo também a aplicação em sistemas produtivos dispersos. Algumas funções disponíveis na plataforma JADE são:

- (i) a *Graphical User Interface* (GUI), que permite monitorar e controlar o estado dos agentes para, por exemplo, parar a execução ou reiniciar as atividades dos agentes;
- (ii) o agente *dummy*, que permite editar, compor, enviar, receber e visualizar mensagens ACL;
- (iii) o agente *sniffer*, que permite rastrear mensagens trocadas entre os agentes usando uma notação baseada em diagramas UML; e
- (iv) o agente *introspector*, que permite monitorar e controlar o ciclo de vida de agentes em execução, as suas mensagens e os seus comportamentos.

Caire, Gotta e Banzi (2008) propõem uma extensão ao JADE, chamada *Workflow Agent Development Environment* (WADE) a qual segue as especificações FIPA. A Figura 15 ilustra os componentes específicos desta plataforma. Os componentes WADE (destacados em azul) fornecem uma extensão que permite criar o *Workflow Engine Agent* (WEA) (CAIRE et al., 2008). Este mecanismo permite desenvolver o WEA com base nos componentes de JADE (destacados em amarelo). Além do comportamento normal de uma plataforma JADE, todo WEA ativo é capaz de executar um fluxo de tarefas de acordo com metamodelos

baseado em *XML Process Definition Language* (XPDL), um padrão especificado pelo *Workflow Management Consortium*. Desse modo, o desenvolvimento de um WEA pode ser feito através de interfaces gráficas de um fluxo de tarefas que está em sincronia com a execução de um programa em código Java para a geração automática do mesmo (CAIRE et al., 2008; CAIRE; GOTTA; BANZI, 2008). WADE disponibiliza uma interface baseada em *Web Services Description Language* (WSDL) (Figura 16) que permite aos agentes de plataformas diferentes comunicar-se e solicitar serviços entre si.

Figura 15 - A plataforma JADE+WADE usada para implementar sistemas multiagentes usando *workflow*.

Fonte: (CAIRE; GOTTA; BANZI, 2008)

Cada instância de JADE+WADE é chamada de *container* e um conjunto deles é chamado de *platform*. Neste caso, o *main container* deve sempre ser ativo assim como os outros *containers* registram-se com ele na inicialização. Uma ou mais aplicações de agentes podem ser iniciadas em um *container* e para executar suas tarefas um agente deve comunicar-se com outros agentes através do formato *Agent Communication Language* (ACL) que é definida pela FIPA (FIPA, 2002). JADE é uma plataforma reconhecida na implementação de sistemas multiagentes e sua estrutura pode ser facilmente acessada em Bellifemine, Caire e Greenwood (2007), enquanto que os componentes específicos de WADE são:

- (i) *boot daemon*: existe um para cada *host* na *platform* e ele ativa os *containers* no seu *local host*;
- (ii) *ConFiguration Agent* (CFA): ele executa no *main container* e é responsável por interagir com os *boot daemons* e controlar o ciclo de vida da aplicação;
- (iii) *Controller Agent* (CA): existe um para cada *container* na *platform* e eles são responsáveis

Figura 16 - Interface para *web Services* de WADE.

Fonte: (CAIRE; GOTTA; BANZI, 2008)

por supervisionar as atividades no *local container* e por mecanismos de tolerância a falhas da plataforma.

2.3.3 Ontologia para HMAS

A troca de mensagens bem como a definição de componentes do controle de sistemas de manufatura exigem um protocolo para adequar a sintaxe e semântica entre eles. A sintaxe define a estrutura de uma linguagem, isto é, uma gramática tipicamente em forma de regras que governam a estrutura das sentenças. Semântica trata dos aspectos de significado como expressado na linguagem, isto é, o sentido dos elementos da linguagem e a combinação delas, incluindo a relação destes elementos ao mundo real. As semânticas são capturadas pela ontologia (OBITKO et al., 2010).

A ontologia é reduzida a uma representação de uma parte de certo domínio. Esta representação é processável pela máquina e é apropriada para aplicações específicas. Neste contexto, ontologia é definida como uma especificação formal de uma visão compartilhada de como representar o mundo real e consiste de conceitos que representam classes de objetos no mundo real, seus atributos, relações e restrições. De fato, a ontologia fornece um vocabulário estático baseado em padrões gerais que são observáveis em tal realidade. Este vocabulário é usado para descrever o estado atual observado do sistema real e sua dinâmica.

A representação é composta de instâncias de classes pré-definidas e é usualmente chamada de base de conhecimento (OBITKO et al., 2010; BÉHÉ et al., 2014). Em outras palavras, ontologia pode ser definida como uma representação de dados de um conjunto de conceitos dentro de um domínio e os relacionamentos entre eles.

Em Lepuschitz et al. (2015) é apresentado um levantamento de ontologias para sistemas multiagentes aplicados à manufatura, com objetivo de avaliar as funcionalidades de um controle de SRM de forma integrada. O objetivo de Lepuschitz et al. (2015) é desenvolver uma semântica extensiva para integração horizontal e vertical e interoperabilidades de dados entre componentes virtuais e físicos. De acordo com estes autores, a representação proposta tem uma estrutura consistente para ser acessada pelos demais componentes fornecendo serviços como análise de desempenho ou gerenciamento de planos de produção. Os autores analisaram e apresentaram várias normas e propostas de ontologia e concluem que nenhuma das representações é suficiente para individualmente atender todos os requisitos de um HMAS para manufatura. Entretanto, é possível obter uma solução combinando os conceitos adotados pelas diferentes propostas.

A ontologia de ADACOR envolve conceitos para componentes de manufatura com recursos físicos (máquinas, ferramentas, etc.) bem como para ordens de produção, operações e distúrbios (LEITÃO; RESTIVO, 2006). A ontologia MASON (LEMAIGNAN et al., 2006) centra-se em aspectos da mesma forma. Assim, estes autores representam uma ontologia para níveis de controle superior do sistema de manufatura.

A utilização de ontologias também tem sido associada a arquiteturas de sistemas baseadas em *Web Services* (WebS). A motivação é estabelecer um *framework* comum onde dados são compartilhados e reutilizados. Trata-se de uma extensão que inclui a descrição semântica de domínios que podem ser encontrados, processados e integrados por *software* e baseado em componentes autônomos distribuídos (isto é, agentes). Estas tecnologias podem ser classificadas em *Resource Description Framework* (RDF) e *Web Ontology Language* (OWL). RDF é um padrão para descrever dados estruturados em forma de simples declarações (MILLER, 1998). Cada declaração é descrita com uma tripla sujeito-predicado-objeto e cada elemento da tripla é identificado por um *Uniform Resource Identifier* (URI). No lugar do objeto, pode ser usado um literal, tais como um *string* ou um número. O conjunto pode ser acionado por recursos distribuídos na internet. Para pesquisar um dado dentro de um gráfico RDF existe o protocolo *Simple Protocol and RDF Query Language* (SPARQL) (PÉREZ; ARENAS; GUTIERREZ, 2006).

A *Web Ontology Language* (OWL) (OBITKO; MAŘÍK, 2003; BECHHOFFER, 2009) é usada para descrição de ontologias e envolve uma semântica em um domínio particular em forma de classes, também chamado de *Terminological boxes* (T-boxes), a relação entre as classes e

as restrições de uso. Assim, a ontologia está associada a uma linguagem que é aplicada para dar o significado de um contexto semântico da representação do mundo real. A informação é armazenada em uma base de conhecimento em forma de instâncias (chamadas de A-boxes ou *assertions*) das classes e relações definidas na ontologia.

WebS foi concebido no contexto de um *framework* para publicar, descobrir e executar remotamente serviços disponibilizados na internet. Para garantir interoperabilidade sintática das WebSs, define-se especificações, incluindo protocolo de mensagens, linguagem para descrição da interface de serviços, protocolos para registro e pesquisa de serviços. Um cenário típico é a decomposição de tarefas complexas em vários serviços em paralelo e/ou em série, e onde os dados resultantes de um serviço são usados como entrada para outro serviço ou onde os dados de um ou mais serviços são combinados para criar uma resposta para a requisição original.

As normas ANSI/ISA-95 e ANSI/ISA-88 também servem de base para definição de ontologias. A ISA-95 descreve modelos hierárquicos da organização da produção e o fluxo de eventos e fornece os conceitos básicos para a integração de sistemas de controle com os processos de negócio do empreendimento. A ISA-88 "*batch control*" trata com mais detalhes processos em batelada do ambiente de produção. Ela define modelos hierárquicos de sistemas de produção do empreendimento, baseados em módulos de controle. Ela também descreve modelos de processo para fases sequenciadas de produção e ações. Outros esforços de normatização tem sido concentrado no *Open ObjectOriented kNowledge Economy for Intelligent inDustrial Automation* (O3NEIDA), um consórcio que visa criar uma infraestrutura aberta de automação baseada em componentes (VYATKIN; CHRISTENSEN; LASTRA, 2005). Lemaignan et al. (2006) afirmam que um mapeamento tem sido desenvolvido entre a ontologia OWL e o modelo interno usado na plataforma JADE.

2.3.4 Outros trabalhos correlatos

Além dos trabalhos já citados, existem outros ainda trabalhos correlatos envolvendo HMAS.

Van Brussel et al. (1998) apresentam um tipo de arquitetura de referência denominada como *Product Resource Order Staff Architecture* (PROSA), que é baseada em quatro tipos básicos de holons: *product*, *resource*, *order* e *staff*. O conceito de agregação é usado para agrupar holons básicos em holons mais complexos (holarquias), e o conceito de especialização serve para especificar holons com base nos holons existentes, tais como os holons *workstation* e *equipment*. Na PROSA, para toda ordem de cliente, um *order* holon é instanciado com base em um *template*, podendo conter o número e tipo de produto, fluxo de produção e restrições temporais. Um *order holon* controla o estado do pedido, acompanhando sua evolução, desde

sua solicitação na produção até a obtenção do produto final. Os *product* holons recebem *requests*⁹ para informar quanto à disponibilidade de recursos da ordem de produção e as etapas para obtenção de cada produto, isto é, quantidade de matéria prima, operações e restrições temporais. Em seguida, os *product* holons enviam *requests* para as várias operações necessárias para fabricar cada produto e recebem as respostas (*bids*¹⁰), como por exemplo, se um robô está livre para carregar *pallets* ou carregar/descarregar peças em uma máquina. Os conceitos adotados na PROSA estão presentes na maioria dos trabalhos em que holons (ou agentes) representam as entidades de um sistema de manufatura.

No trabalho de Chirn e McFarlane (2000) descreve-se uma arquitetura baseada em componentes denominada de *holonic component-based architecture*. Os objetos físicos de um sistema produtivo são caracterizados em dois grupos de acordo com suas propriedades: recursos – que realizam as operações produtivas, e item/produto – que recebem as operações/atividades produtivas. Desse modo, definem-se os holons de recursos que são os elementos responsáveis pela fabricação, montagem, transporte e testes de itens (materiais e/ou informações). De forma similar, definem-se os holons de produtos ou componentes de item/produto cuja parte física são as peças, matéria-prima, *pallets* e a parte de controle que contém as informações sobre as rotas, o processo, a tomada de decisões e os dados de produção.

Cavalieri et al. (2000) propõem incluir um nível supervisor a um sistema holônico para aumentar seu desempenho. Ou seja, os autores propõem uma arquitetura hierárquica parcial dinâmica. Neste mesmo sentido, Leitão e Restivo (2006) propõem *ADaptive holonic COntrol aRchitecture* (ADACOR) que tem uma classe de holon que é responsável por tarefas de supervisão de outros holons autônomos e cooperativos os quais representam recursos físicos (tais como uma máquina CNC ou um robô) ou uma entidade lógica (como uma ordem). São quatro tipos básicos de holons: *product*, *task*, *operational* e *supervisor*. Os três primeiros tipos são similares aos holons *product*, *order* e *resource* da PROSA, e o *supervisor holon* é similar ao *staff* holon, sendo que neste são incluídos planos de supervisão e otimização da programação para realizar as tarefas operacionais. Em Pach et al. (2012), ADACOR é aplicado na alocação dinâmica de máquinas e roteamento dos produtos em sistemas de manufatura e o *software* Netlogo é usado na validação dos modelos.

Vrba et al. (2011) utilizam o conceito de estação de trabalho, isto é, uma abstração de um grupo de máquinas, equipamentos e ferramentas, que realizam várias operações de produção e que também são vistos como recursos. Os autores propõem encapsulamentos da parte do alto nível de controle, do subsistema do baixo nível de controle, e da interface de controle para

⁹Termo utilizado na comunicação entre agentes baseado nas especificações da FIPA (FIPA, 2002).

¹⁰Termo utilizado na comunicação entre agentes baseado no CNP (SMITH, 1980).

interações entre esses níveis. É proposta uma forma de implementar a interface de controle e compartilhar os dados diretamente na memória de um controlador industrial programável tornando possível a criação dinâmica dos agentes nestes controladores industriais. A interface de controle capacita os agentes a ler e escrever os valores de *tags* nos controladores industriais e também subscrever os valores de *tags* modificados. Por exemplo, um agente pode definir uma *tag* genérica I/O de uma válvula como *close*. Se existem duas válvulas V1 e V2, seus atributos têm que ser armazenados em dois diferentes *tags*, como *V1_close* e *V2_close*. A interface de controle pode ser tanto em C++ quanto em Java, e para acesso remoto às *tags*, os autores utilizam *object linking and embedding for process control* (ZHENG; NAKAGAWA, 2002) e uma implementação proprietária da Rockwell.

Hsieh (2009) estuda os desafios e mecanismos para implementar a reconfiguração em HMAS. O autor propõe ferramentas e uma reconfiguração de sistemas holônicos considerando restrições temporais e usando o CNP e a RdP. Mudanças e distúrbios em HMAS são apresentados, considerando a otimização de processos produtivos.

Chen e Occeña (2000) apresentam a modelagem de projeto de produto, considerando o conhecimento de diferentes especialistas de forma simultânea. O método “quadro-negro” (HAYES-ROTH, 1985) para comunicação de sistemas multiagentes é usado para gerenciar os diferentes atores envolvidos no projeto. A arquitetura baseada em quadro-negro foi um dos primeiros métodos de interação usado em sistemas multiagentes. Neste método o agente sobrescreve, lê e troca informações de forma indireta e coordenada via uma estrutura comum (quadro-negro) de armazenamento, planejamento e disponibilização de informação ou tarefas. A ideia do método é facilitar a troca de conhecimentos entre os agentes. Atualmente a função do quadro-negro, tal como em JADE, é substituída pelo diretório de páginas amarelas (*directory facilitator*).

Maione e Naso (2003) propõem um sistema de produção no qual produtos e recursos têm capacidade de decisão. Os agentes que representam eles podem comunicar entre si, tomar decisões para atingir seus objetivos e estabelecer um plano de produção.

DeLoach e Garcia-Ojeda (2014) propõem a *Multi-agent System Engineering* (MaSE) que fornece soluções práticas para HMAS usando um *software* dividido em fases de análise e projeto. A *Methodology for Ontology-Based Multi-Agent System* (MOBMAS) proposta por Tran e Low (2008) é desenvolvida através de atividades interativas e sequenciais. Cernuzzi, Cossentino e Zambonelli (2005) propõem o *Plant Automation BAsed on DIstributed Systems* (PABADIS) que é uma metodologia passo-a-passo considerando do requisito ao código para projetar e desenvolver um sistema multiagente, integrando modelos de projeto e conceitos de engenharia de *software* orientada a objetos. ANEMONA (BOTTI; BOGGINO, 2008) é uma metodologia baseada em desenvolvimento que combina as abordagens *top-down* e *bottom-up*

dividindo o sistema em aspectos específicos sob diferentes visões do sistema.

Shen et al. (2007) propõem um sistema multiagente com integração orientada a serviço para programar empresas virtuais. Uma ontologia fornece a semântica para integração dos serviços, considerando resposta e requisições, técnicas de raciocínio da ontologia e formação de parcerias.

Uma abordagem para definir a especificação de processos alinhada com o contexto de *cyber-physical systems* é apresentado em Leitão et al. (2015). Esta abordagem integra conceitos de SOA e HMAS considerando uma estratégia baseada em produtos inteligentes, orquestrando serviços fornecidos por agentes associadas à dinâmica das operações de cada produto e seu controle de qualidade

Quintanilla et al. (2015) consideram outra combinação do uso de SOA em HMAS aplicada a sistemas do tipo *job-shop* dividido em *workstations*.

Sallez, Berger e Trentesaux (2009) utilizam Netlogo para determinar dinamicamente a melhor rota para produtos em situações caracterizadas pela ocorrência de distúrbios.

Estes trabalhos confirmam o potencial de aplicação de HMAS que pode ser explorado de diferentes formas.

2.4 ARQUITETURA ORIENTADA A SERVIÇOS

A abordagem de *Service-Oriented Architecture* (SOA) considera entidades colaborativas, caracterizadas pela modularidade e reconfigurabilidade no controle de sistema de manufatura (MELO et al., 2008; MELO et al., 2009; FATTORI et al., 2011; QUINTANILLA et al., 2015). SOA envolve um conceito abstrato de uma arquitetura de *software* baseada no encapsulamento de funcionalidades de recursos como serviços que podem ser oferecidos, pesquisados e usados por outras entidades que requisitam os serviços (MELZER, 2010). Os princípios da SOA consideram o reuso de componentes colaborativos, distribuídos e fracamente acoplados.

De acordo com Fugita e Hirama (2012), SOA estabelece um paradigma para construção e integração do *software* que estrutura aplicações em elementos modulares chamados serviços. O serviço, a unidade fundamental de uma SOA, é um elemento computacional que tem como propósito desempenhar uma função específica e que pode ser utilizado por um cliente. É o resultado de uma ou mais atividades dentro do contexto do desenvolvimento de um *software* ou sistema. Na SOA, artefatos são definidos como produtos de tarefas realizadas, as quais por sua vez podem utilizar, gerar ou modificar os artefatos.

Em geral, quando SOA é aplicada a sistemas de manufatura entende-se que na arquitetura de controle é colocada uma camada de integração entre as camadas 2 e 3 da Figura 3 (ANSI/ISA, 1995), sendo essa um *middleware* que permite a integração de múltiplos serviços em camadas

diferentes (GROBA et al., 2008).

No contexto de *software*, ao projetar um sistema de controle baseado em SOA deve-se descrever as fases de *orquestração* e *coreografia* de serviços. *Orquestração* refere-se à interação dos serviços para composição de processos específicos enquanto que *coreografia* (composição) descreve as sequências de interação entre múltiplos processos (PELTZ, 2003).

Para descrição da interface de comunicação dos serviços uma das tecnologias mais utilizadas é o WSDL. Um WebS é concebido para permitir o acesso consistente a recursos computacionais em múltiplas plataformas heterogêneas, estabelecendo padrões de interoperação entre diferentes aplicações de *software* (FARAHBOD, 2001). SOA e WebS têm uma semelhança em relação a seus atores, pois ambos têm como princípio o baixo acoplamento (*loose coupled*) entre as partes envolvidas e alta granularidade (*fine grained*) na definição.

Portanto, a SOA e seus procedimentos de interação tem sido proposta para assegurar a integração entre sistemas heterogêneos de empreendimentos dispersos e, além disso, sua combinação com técnicas de HMAS também tem sido investigada. Mendes et al. (2009) utilizam agentes para alocar recursos na forma de serviços e Quintanilla et al. (2015) usam conceitos de HMAS para desenvolver um sistema produtivo com operações sequenciais e rotas flexíveis e onde processos das *workstations* têm um fraco acoplamento entre si, porém necessitam de uma sincronização. A extensão WADE também aplica conceitos de SOA propondo através de *Web Services Description Language* (WSDL) rotinas para os agentes requisitarem serviços. Além disso, WADE fornece uma extensão chamada *Workflow Engine Agent* (WEA), que possui uma ferramenta baseada em *workflow* para modelar agentes do tipo JADE. Na Figura 16, está ilustrada a interface de WSDL de WADE para o agente que requisita um serviço (como cliente) ou aguarda até que um serviço é invocado (servidor). A interface permite especificar a forma de importação: arquivo ou através de *Uniform Resource Locator* (URL). Assim, agentes podem requerer os serviços disponíveis pelos agentes de outras *platforms*. A Figura 16 ilustra a interface onde o desenvolvedor pode programar um agente para ele requerer um serviço seja como cliente ou como servidor.

Serviços são similares aos componentes em muitos aspectos, por exemplo, ambos consideram a ênfase em interfaces. Contudo, serviço é a representação lógica de uma tarefa de negócio repetitiva que tem um resultado específico, por exemplo, verificação de disponibilidade de uma máquina e fornecer dados de entrega de um produto. SOA estabelece a orientação a serviços, com foco nos processos de negócios e HMAS fornece uma base para a implementação de SOA. Pode-se considerar que serviços encapsulam componentes HMAS e a SOA tem foco nas interfaces de forma a trabalhar com acoplamento fraco, composição, reuso e vários outros objetivos do projeto de um sistema de manufatura. Além disso, SOA deve atender o negócio rapidamente e, para tanto, obter um nível de granularidade para os serviços identificáveis

nos processos de negócio é uma abordagem fundamental. Isto contribui principalmente em questões como a orquestração no contexto de *business process management*, uma vez que permite a visibilidade da associação do serviço com o negócio. Sistemas de informação em geral possuem a lógica de processo de negócio e a lógica de integração misturada com as regras do negócio. Um grande desafio na abordagem SOA é dividir corretamente a lógica da aplicação.

2.5 DISCUSSÃO

As seções anteriores descrevem os conceitos fundamentais a serem considerados para a concepção de uma solução de controle para SRMs. São levantadas as características necessárias para projeto do sistema de controle de um sistema reconfigurável de manufatura, isto é: sistema distribuído, autonomia, inteligência, tolerância a falhas, adaptabilidade e reuso de modelos. Também são consideradas as técnicas formais para modelagem e análise de sistemas de controle.

A revisão bibliográfica destacou diversas características requeridas para projetar o controle de um SRM e as ferramentas disponíveis para a implementação de sistemas de controle. As características principais das metodologias, arquiteturas e aplicações bem como tendências e desafios relacionados a sistemas de controle de manufatura foram também apresentados. Algumas propostas têm visão orientada a projeto do produto e focam na escalabilidade de componentes genéricos dos sistemas (HSIEH; CHIANG, 2011). Outras têm uma visão orientada a projeto de máquinas e focam mais na otimização da utilização da máquina do que no desempenho do plano de produção (BARBOSA et al., 2012). Entretanto, o controle de um SRM deve ser responsável por negociar e planejar dinamicamente as atividades realizadas na planta com base nos pedidos feitos por clientes, alterações destes pedidos, ocorrência de problemas nos fornecedores e falhas nos equipamentos. Nessa negociação existe o fluxo de informações de capacidade de produção, custos, materiais, equipamentos, recursos humanos, e informações quanto à viabilidade de tempo e prazo de entrega. Além disso, considerando a dispersão dos sistemas envolvidos, os componentes devem considerar a interoperabilidade em ambientes heterogêneos.

Uma alternativa é considerar a abordagem de SOA com suas entidades colaborativas, características de modularidade e reconfigurabilidade no controle de sistema de manufatura (MELO et al., 2008; MELO et al., 2009; FATTORI et al., 2011; QUINTANILLA et al., 2015). De acordo com Candido et al. (2010), SOA tem revolucionado a computação distribuída como uma solução para a colaboração inter-empresarial, mas, embora forneça o *framework* para integrar serviços de diversas companhias considerando interoperabilidade entre elas, SOA

não apresenta métodos formais que garantam uma sistemática apresentação das especificações de projeto.

Outro importante conceito para o controle de um SRM é a distribuição de tarefas em diferentes componentes autônomos. Neste sentido, a adoção de técnicas e conceitos relacionados que permitam a combinação entre HMAS e SOA permite desenvolver um sistema decompondo este em partes de forma que cada um possa executar serviços básicos e que são posteriormente combinados para atender o objetivo global do sistema. A extensão WADE (CAIRE; GOTTA; BANZI, 2008) para a plataforma de desenvolvimento de HMAS JADE (BELLIFEMINE; CAIRE; GREENWOOD, 2007) aplica conceitos de SOA propondo através de rotinas a disponibilidade que os agentes requisitem serviços. Além disso, WADE possui um módulo baseado em *workflow* para o projetista desenvolver de forma gráfica as aplicações baseadas em agentes JADE.

Relacionado à modelagem de sistemas de manufatura e das funções de controle envolvidas, observa-se que o PFS associado à IOPT é uma solução efetiva para a construção de modelos desenvolvidos de forma sistemática e modular. Estes modelos, por sua vez, podem ser desenvolvidos com a IOPT *web tool* (GOMES et al., 2007; PEREIRA et al., 2012), a qual está disponível na internet e permite transformar de forma automática os modelos em programas de controladores industriais. Utilizando esta ferramenta as propriedades estruturais e comportamentais podem ser examinadas nos modelos resultantes de modo que as soluções de controle podem ser verificadas e validadas, assim como análise de desempenho podem ser conduzidas. Portanto, o PFS e a IOPT comprovam ser adequados para modelagem do controle de sistema reconfigurável de manufatura.

O levantamento apontou ainda a falta de métodos apropriados e técnicas para facilitar e sistematizar a implementação de HMAS/SOA. Há falta de dados quantitativos sobre implementações práticas e seus mecanismos de decisão e autonomia. A abordagem usual é a separação em agentes da camada de alto nível de controle e tomada de decisões, executados em computadores, e em agentes da camada de baixo nível, executados nos controladores industriais. Assim, tem-se os agentes escritos em uma linguagem orientada a objeto, tal como *C++* ou Java, e os agentes da camada de baixo nível em linguagens que não são orientadas a objeto, tal como *Sequential Flow Chart* (SFC) que é muito similar a RdP. Isto porque os controladores industriais estão sujeitos a restrições técnicas e não consideram os métodos propostos em HMAS/SOA, de forma que os programas não possuem características como herança, instanciação, ou virtualização. Para análise detalhada de algumas variáveis do sistema, Vrba et al. (2011) apresentam um desenvolvimento neste sentido; entretanto a solução é proprietária.

Um HMAS deve conceitualmente apresentar uma estrutura de comunicação que garante

o alto nível de autonomia dos holons e funções baseadas essencialmente no comportamento que devem ser determinados através de cooperação com outros holons, ao invés de serem determinados por um mecanismo centralizado. Portanto, um mecanismo de negociação é mais favorável do que um mecanismo de comunicação de solicitação-resposta entre holons.

Outro desafio é o desenvolvimento de um mecanismo de controle para que o HMAS atinja os objetivos necessariamente dentro de um prazo requerido, ou seja, que forme holarquias considerando os prazos de produção (*lead time*) requeridos. Além disso, cada subsistema de manufatura deve ser controlado por supervisores responsável pela condução das atividades produtivas de acordo com os planos previstos e pelo diagnóstico e tratamento de falhas. Ou seja, o supervisor deve ter autonomia para aplicar a reconfiguração para ajustar um SRM para diferentes níveis de produção na ocorrência de falhas ou novas demandas derivadas de eventos imprevistos.

Para desenvolver um HMAS, uma ontologia deve representar um conjunto de conceitos dentro de um domínio e os relacionamentos entre eles. A maioria das ontologias são baseadas em padrões como ANSI/ISA 88, O3NEIDA ou MASON. Entretanto, nenhuma delas fornece um modelo semântico para completar a extensibilidade e empregabilidade em controle de SRM que considere os novos paradigmas de produção, isto é, manipular ordens personalizadas, material e produtos intermediários, transporte entre máquinas de produção e sua descrição. ANSI/ISA 88 é concebida para processos industriais do tipo batelada, O3NEIDA não fornece o que é necessário para o problema identificado e MASON é indicada para sistema multiagente porém contém fatores limitantes a aplicações específicas. A ontologia deve ser o mais geral possível, de modo a fornecer uma descrição consistente do sistema, na qual extensões de ontologias específicas podem ser adotadas (OBITKO et al., 2010). A ontologia deve ser relativamente simples de forma ser aplicável em sistemas reais. Dos estudos citados anteriormente, é possível verificar características interessantes em ontologias existentes. Além disso, o projeto do sistema de controle deve ser integrado à ontologia que por sua vez deve ser definida desde a concepção do mesmo.

Relacionado à análise de um HMAS, observa-se que alguns trabalhos apresentam certa complexidade para testes via simulação, pois os *software* são desenvolvidos especificamente para cada caso, isto é, requerem significativo esforço para analisar o HMAS em outras aplicações. Esta complexidade é evidentemente maior se for considerado os mecanismos de reconfiguração que permitam melhor adaptação a mudanças no ambiente. Assim, o método para projetar HMAS em diferentes domínios de sistemas de manufatura é fundamental e entende-se que o desenvolvimento de uma plataforma específica não atende a este objetivo. Portanto, o Netlogo é considerado aqui como a ferramenta a ser explorada visto que apresentou melhores resultados em estudos comparativos (BARBOSA et al., 2011; CASTLE; CROOKS,

2006), destacando-se sua disponibilidade de suporte/documentação, facilidade de programação e interface amigável.

Com base no levantamento realizado, introduz-se no próximo capítulo que é possível definir uma arquitetura baseada em HMAS e SOA na qual sociedades de agentes são organizadas de forma modular e seus modelos são desenvolvidos usando as técnicas do PFS, IOPT e ferramentas disponíveis.

3 ARQUITETURA DE CONTROLE PROPOSTA

3.1 WORKFLOW PARA A CONCEPÇÃO DA ARQUITETURA

O *workflow* que deu origem à arquitetura proposta para o controle de sistemas reconfiguráveis de manufatura pode ser ilustrado de acordo com a Figura 17. Neste *workflow*, para criar e modificar processos produtivos, a engenharia de sistemas define modelos de certo domínio de aplicação, isto é, abstrai a estrutura e a operação do sistema real a partir de métodos adequados. Por meio da arquitetura de controle especificam-se os componentes deste sistema, suas funções e a interação entre eles. No presente trabalho, considera-se que esta especificação resulta em holarquias, de acordo com o conceito de sistemas holônicos (COLOMBO; NEUBERT; SCHOOP, 2001). Desse modo, se concebem “artefatos”¹¹ que devem ser implementados ou modificados com a participação de gestores e desenvolvedores do sistema, que são as pessoas comprometidas e capacitadas para esta fase de desenvolvimento. O envolvimento dessas pessoas é fundamental para a definição do controle de sistemas de manufatura com capacidade de reconfiguração. Entende-se aqui que os métodos para desenvolvimento e especificação dos artefatos também deve ser aplicados na definição da arquitetura de controle proposta.

Figura 17 - *Workflow* do desenvolvimento da arquitetura de controle proposta.

Na arquitetura de controle de um sistema de manufatura reconfigurável, quando um processo produtivo solicita um recurso, na verdade, está solicitando aos holons para verificar as funcionalidades requeridas, escolhendo aquela que, de acordo com o plano de produção requerido, atenda aos objetivos planejados.

¹¹Em projeto de SOA, artefatos são definidos como itens resultantes de tarefas. Artefatos devem ser bem definidos, tangíveis e podem ser compostos de outros artefatos. As tarefas podem utilizar, gerar ou modificar os artefatos (ERL, 2005; FUGITA; HIRAMA, 2012).

Para o controle supervisorio do sistema de manufatura reconfigurável, a implementação das funções de controle é feita em linguagem Java usando JADE (BELLIFEMINE; POGGI; RIMASSA, 1999) e sua extensão WADE (CAIRE; GOTTA; BANZI, 2008). Eles formam uma plataforma de código aberto para o desenvolvimento de sistemas multiagentes e onde o conhecimento não é alocado em um único servidor, sendo distribuído para aumentar a robustez do sistema. Considera-se que mesmo as soluções proprietárias dos dispositivos de chão-de-fábrica devem ser passíveis de serem transformadas em programas computacionais escritas em linguagem unificada *eXtensible Markup Language* (XML), o que garante também a atualização de sistemas legados. Além disso, para assegurar a colaboração dos holons da arquitetura proposta, considera-se os conceitos de SOA que estão previstos na extensão WADE para integrar sistemas heterogêneos.

3.2 COMPOSIÇÃO DAS CLASSES

De acordo com o ponto de vista do projetista, diferentes classes de holon podem formar holarquias que podem ser consideradas como um conjunto de holons que colaboram entre si e que também podem ser decompostos em holons atômicos, os quais não podem ser mais decompostos (GIRET; BOTTI, 2009). Neste trabalho, a arquitetura de controle é considerada composta pelas seguintes classes de holons (Figura 18):

- (i) holon de produto – *Product Holon* (PrH) – que representa a matéria prima, produtos intermediários e os produtos finais. O conjunto de PrHs forma a holarquia dos planos de produção. Esta representa um fluxo envolvendo a combinação de matéria-prima, produtos intermediários e as operações (tais como, usinagem e montagem) executadas para obter os produtos finais. Ou seja, cada PrH tem sua sequência de processos que envolvem operações para obtenção dos respectivos produtos que ele representa;
- (ii) holon de estratégia – *Task Holon* (TH) – que representa o *Business Process Management* (BPM), que são as estratégias planejadas para atender uma ordem de cliente, monitorando e requisitando serviços disponíveis pelos outros holons/holarquias. O TH também é concebido para representar as estratégias de reconfiguração;
- (iii) holon supervisor – *Supervisor Holon* (SuH) – que contém o conhecimento necessário para coordenar as operações que compõem os processos produtivos, registrando a inclusão de novos holons de acordo com as habilidades de cada componente. Além disso, o SuH verifica o comportamento do sistema, coordena e muda a holarquia de acordo com condições específicas, gerando composição de holarquias organizadas. Ou seja, de acordo com os objetivos, habilidades, estados ou ocorrência de falhas, o SuH organiza

Figura 18 - Arquitetura e holarquias representando os níveis de controle em PFS.

novas conexões; adiciona ou remove eventos, entradas e saídas; e atualiza algoritmos, e, desse modo, coordena a reconfiguração do sistema de manufatura. As trocas de mensagens são também monitoradas pelos SuHs e representadas na Figura 18 por elementos `distribuidores` do PFS; e

- (iv) holon operacional – *Operational Holon* (OpH) – que representa os recursos físicos os quais têm dispositivos de controle para sua operação automática, equipe de produção e operadores. De acordo com a conveniência, um ou mais dispositivos podem ser representados por um único OpH. Assim, cada OpH representa um serviço atômico o qual pode ser combinado com outros serviços atômicos compondo holarquias que representam os processos produtivos e assim por diante até representar todo o controle de sistema reconfigurável de manufatura.

Para representar componentes e funcionalidades desde o nível de controle corporativo até dispositivos de controle supervisor do chão de fábrica, a recursividade proporcionada pela técnica de sistemas holônicos é combinada com a representação da hierarquia de processos oferecida pelo PFS da seguinte forma:

- (i) um holon pode representar todo um controle de SRM e ao mesmo tempo ser decomposto em outros holons até um nível atômico.
- (ii) um holon pode interagir com outros holons para atingir objetivos, formando holarquias;
- (iii) elementos do PFS são usados para representar holons/holarquias, os quais representam os processos produtivos e podem integrar componentes no mesmo nível de controle ou de níveis diferentes.

Por exemplo, na Figura 18 é mostrado um exemplo onde um holon que representa o controle de SRM é decomposto em outros holons, os quais são representados por elementos do PFS. Tem-se `atividades` do nível corporativo representando processos relacionados a gerenciamento do negócio (BPM) e processos relacionados a reconfiguração. Estas `atividades` são refinadas para representar processos nos demais níveis de controle até obter-se a representação de processos relacionadas ao objeto de controle no nível de controle supervisor. Além disso, tem-se representação de conceitos relacionados a SOA e sistemas dispersos, tais como a representação da interação entre os holons (“orquestração”), a supervisão e o compartilhamento de serviços. A orquestração de holarquias pode ocorrer de duas formas fundamentais:

- (i) por meio da composição de processos, que se baseia na representação de processos de negócios; ou

- (ii) por meio da representação de funcionalidades, que se baseia na representação de especificidades de operadores e dispositivos.

Neste contexto, holons (ou holarquias) podem interagir no mesmo nível e com holons (ou holarquias) de outros níveis de controle. As interações podem variar de uma simples troca de informações e requisições para execução de operações até negociações relativamente mais complexas, usando protocolos específicos. Por exemplo, os holons de uma holarquia que representa os processos produtivos do nível corporativo, enviam requisitos de plano de produção para o nível de fábrica, o qual, por sua vez, envolve a composição de supervisão de serviços do nível de controle supervisorio de holons que representam operadores e objeto de controle.

Para realizar a orquestração, cada holon tem uma interface pré-definida, a qual é gerida pelo SuH e representada por um elemento `distribuidor`. Cada `distribuidor` é identificado por “i_nome do primeiro holon_nome do segundo holon” de acordo com o nome dos holons que podem atender determinada conexão. Esta identificação em conjunto com outros mecanismos e a definição da sequência de operações de cada processo permitem uma alocação de recursos considerando a funcionalidade de cada holon. Além disso, em caso de redundância de funcionalidades onde mais de um holon pode completar uma ação, é possível realizar a conexão de interfaces com diferentes holarquias, permitindo assim maior flexibilidade de reconfiguração ao sistema. A Figura 19 representa um exemplo onde alguns modelos formam diferentes holarquias e onde a conexão entre PrHs pode ser feita de duas formas diferentes conforme ilustrado no detalhamento do `distribuidor`. Ou seja, neste caso a conexão pode ser feita por `i_PrH-1_PrH-2` que conecta PrH-1 com PrH-2 e `i_PrH-1_PrH-3` que conecta PrH-1 com PrH-3. Esta figura também representa composição de holarquias entre outros holons e ilustra SuHs que são responsáveis pela coordenação da orquestração entre eles.

Considerando os estudos da Subseção 2.3 e as referências citada por eles, na Tabela 5 é apresentado um comparativo das diferentes abordagens e denominações baseado nos estudos anteriores, com a inclusão dos holons propostos aqui. Observa-se que a arquitetura de controle proposta tem similaridades como outras propostas, tais como: o PrH é similar ao proposto por Van Brussel et al. (1998) e o SuH adota mecanismos propostos por Leitão e Restivo (2006). Entretanto, os holons, a composição das holarquias para execução de ordens de produção e a abordagem de modelagem nesta proposta são diferentes. Além disso, esta arquitetura de controle considera as características de reconfiguração propostas por Koren (2014) e outras boas práticas de projeto de sistema de controle como maior interação entre humanos e processos, interoperabilidade e a composição de holarquias baseado em serviços.

Figura 19 - Composição de holarquias em PFS por meio de distribuidores representando interfaces.

3.3 MECANISMOS DE TRATAMENTO DE FALHAS

O tratamento de falhas em um sistema pode ser entendido como um mecanismo com fases de atuação, ilustradas na Figura 20. No contexto da arquitetura de controle proposta se considera que estes mecanismos estão presentes em cada holon independente do nível hierárquico, mas considerando o grau de autonomia e as habilidades de reação de cada um. As fases são as seguintes: estimação (onde é realizado um diagnóstico para identificação da falha), planejamento (onde ocorre a tomada de decisão da ação que deve ser executada), execução (onde é executada a ação de reconfiguração do sistema) e aprendizagem (onde dados relevantes são armazenados para facilitar outras atuações do tratamento de falhas). Portanto,

Tabela 5 - Comparativo entre arquiteturas de controle.

metodologia/ holon	order	product	task/ operati- ons	resource/ machine	supervisor
PROSA (VAN BRUSSEL et al., 1998)		product holon	order holon	resource holon	staff holon
ADACOR (LEITÃO; RESTIVO, 2006)		product holon	task holon	operational holon	supervisor holon
HCBA (CHIRN; MCFARLANE, 2000)	coordinator		supply agent	resource holon	contact agent
Rockwell (VRBA et al., 2011)	order agent	production plan agent	product agent	work station/ equipment agent	
arquitetura de controle proposta	holarquias	product holon	task holon	operational holon	supervisor holon

pode-se afirmar que o controle do SRM atua de acordo com as seguintes regras:

- (i) Se *sintomas* Então *selecione falha*;
- (ii) Se *falha selecionada* Então *selecione ação*;
- (iii) Se *ação selecionada* Então *ative reconfiguração*; e
- (iv) Se *reconfiguração executada* Então *armazene dados relevantes*.

Figura 20 - Fases do tratamento de falha.

Assim, para um sistema tolerante a falhas, deve-se considerar mecanismos de reconfiguração, diagnóstico de falhas e tomada de decisões. No caso das fases de diagnóstico de falhas e de tomada de decisão são adotados os mecanismos: “diagnosticador” e “decisor”. Estes mecanismos permitem a parada e inicialização dos holons de acordo com a necessidade de reconfiguração do sistema. Por exemplo, se um SuH está monitorando um compressor quando

este dispositivo está em operação, consequentemente, o seu diagnosticador está inicializado e subscrito para o repositório processar os dados. Similarmente, o decisor subscreve quando requisitado as possíveis ações que podem ser executadas para resolver uma situação de falha. O operador pode assim requerer os dados em qualquer estágio e tomar decisão ou armazenar os dados relevantes. Assim, por meio da colaboração entre os agentes a falha pode ser detectada em diferentes níveis de granularidade e, pela colaboração entre holons, a robustez do sistema é assegurada com o processamento de dados obtidos através da representação de componentes individuais e da informação fornecida por sensores, representados pelos holons.

No presente caso, o diagnosticador é baseado na proposta de Sampath et al. (1996) que é baseada em modelos de processos em autômatos finitos. Para o decisor adota-se regras de inferência, tipo “se...então...” baseadas em raciocínio (KUIPERS, 1994). Aqui, os modelos são convertidos para RdP, focando na especificação do controle do SRM. Para cada estratégia de controle representado pelo TH, são construídos dois modelos: (i) modelo geral de funcionamento do sistema e (ii) modelo do diagnosticador. Para o modelo do diagnosticador são considerados “eventos observáveis”, aqueles relacionados a comando de estratégia de controle do TH, e “eventos não observáveis”, que podem estar relacionados a falhas.

De acordo com as possíveis causas de falhas indicadas pelo diagnosticador, ações devem ser executadas pelo decisor. Este mecanismo é especificado baseado em regras de tomada de decisão, dados probabilísticos, relações lógicas ou outros formalismos, para inferir, baseado nas possíveis causas, qual o estado real do sistema.

A construção dos mecanismos diagnosticador e decisor é detalhada no Capítulo 4.

3.4 MECANISMOS PARA COMPOSIÇÃO DE HOLARQUIAS

Para processar uma ordem de produção, uma holarquia é formada baseada na colaboração de holons e dinâmica dos processos produtivos. Esta colaboração garante características essenciais para o controle de sistema reconfigurável de manufatura, tais como agilidade para implementar uma reconfiguração. O desafio são os métodos para que sistemas holônicos possam formar a holarquia mais adequada à ordem de produção e recursos disponíveis. Portanto, nesta seção são apresentados mecanismos que são aplicados aos holons da arquitetura para composição de holarquias.

De forma automática, o “mecanismo baseado em custos” permite definir a holarquia usando prazos pré-definidos para os processos. Esta variável pode ser associada a uma característica de interesse do sistema. O projetista deve analisar para o domínio da aplicação e definir a variável que pode melhor representar o custo de cada processo, tais como um valor que represente o

gasto energético do processo ou o tempo de processamento.

Na proposta de Hsieh (2009), esse custo é associado ao tempo de processamento dos processos representados por holon/holarquia. Por exemplo, considere que $[H_1]$ e $[H_2]$ sejam duas sequências de PrHs que representam planos de produção: $[H_1] : [A_0] \rightarrow [A_1] \rightarrow [A_2] \rightarrow [B_0] \rightarrow [B_1] \rightarrow [B_2] \rightarrow [A_2 + B_2]$ e $[H_2] : [A_0] \rightarrow [A_3] \rightarrow [B_0] \rightarrow [B_3] \rightarrow [A_3 + B_3]$. Suponha que o prazo para a entrega de PrH- $[A_1]$ é $\tau_{due}[A_1]$. Este prazo impõe uma restrição temporal sobre PrH- $[A_0]$ na sequência de $[H_1]$. Considere que $\tau_{due}[A_0]$ seja o prazo devido para que o processo relacionado ao produto representado por PrH- $[A_0]$ complete suas operações. Considere que $\Delta\tau_{due}[A_1][task_n]$ seja o intervalo de tempo em que cada tarefa (*task*) “n” de PrH- $[A_1]$ deve completar suas operações. Considere ainda que PrH- $[A_1]$ começa seu processo na tarefa “1” e termina na tarefa “z”. Para atender a restrição temporal, a Equação (1) deve ser satisfeita por PrH- $[A_0]$ e assim por diante para os outros holons. Este mecanismo permite monitorar o processo e intervir no sistema caso o tempo devido ultrapasse certo limite de atraso definido pelo planejamento.

$$\tau_{due}[A_0] \leq \tau_{due}[A_1] - \sum_{n=1}^z \Delta\tau_{due}[A_1][task_n] \quad (1)$$

É possível definir uma variável chamada de “custo” para cada sequência H_n no caso de holarquias ou h_n no caso de holons. Por exemplo, considere que C_{H_n} e C_{h_n} seja o custo de uma sequência “n” de uma H_n ou de um h_n , respectivamente. Considere que c_{task_n} seja o custo de uma tarefa “n”. Para compor holarquias, um controle de SRM deve ser programado para automaticamente calcular a holarquia que oferece o menor custo. Neste exemplo, o sistema de controle deve comparar o custo de C_{H_1} e C_{H_2} para decidir qual é a melhor sequência de produção, de acordo com as Equações (2) e (3). O c_{task_n} para cada PrH é calculado de acordo com o caso para PrH- $[A_1]$ na Equação(4).

$$C_{[H_1]} = C_{[A_0]} + C_{[A_1]} + C_{[A_2]} + C_{[B_0]} + C_{[B_1]} + C_{[B_2]} + C_{[A_2+B_2]} \quad (2)$$

$$C_{[H_2]} = C_{[A_0]} + C_{[A_3]} + C_{[B_0]} + C_{[B_3]} + C_{[A_3+B_3]} \quad (3)$$

$$C_{[A_1]} = c_{task_1} + c_{task_2} + \dots + c_{task_z} \quad (4)$$

Para compor holarquias, os holons colaboram entre si realizando negociações. O “mecanismo de negociação” proposto por Leitão e Restivo (2006) entre holons baseado no *Contract Net Protocol* (SMITH, 1980) é adotado aqui. Este protocolo define regras que permitem negociação entre holons baseada em créditos “recompensas” e débitos “penalidades”, dependendo se uma ordem é executada com sucesso ou não. Quando um TH está encarregado

de executar uma determinada estratégia, ele recebe as seguintes informações do PrH: a estratégia escolhida, uma medida quantitativa chamada “fundo do serviço” (π), o tempo programado, um valor de penalidade pelo atraso (φ), e um valor de recompensa (ε) por finalizar com sucesso. O TH deve gerenciar a negociação com os OpHs para atingir o objetivo sem ultrapassar o valor limite previsto para π . Durante alocação de novas ordens, há interação entre THs e OpHs. Após esta interação, o TH aceita transferir (pagar) uma recompensa ε ao OpH pela conclusão da ordem e receber uma penalidade φ se a ordem não for concluída no tempo devido. O desempenho dos holons é avaliado pelo resultado obtido dos valores obtidos de recompensas menos penalidades associando uma variável (μ) que é atualizada de acordo com a execução das ordens. A Tabela 6 apresenta um exemplo da evolução desse mecanismo.

Tabela 6 - Mecanismo de negociação baseado no CNP (SMITH, 1980).

fase	holon de estratégia	holon operacional
alocação de uma ordem	contrata a execução da ordem por ε e a penalidade por φ	contrata a execução da ordem por ε e a penalidade por φ
finaliza uma operação com sucesso	paga o valor ε ao OpH, i.e., $\pi \leftarrow \pi - \varepsilon$	aumenta o total dos créditos por ε , i.e., $\mu \leftarrow \mu + \varepsilon$
finaliza uma operação com atraso	paga o valor ε e recebe o valor φ do OpH, i.e., $\pi \leftarrow \pi - \varepsilon + \varphi$	diminui o total dos créditos por φ e aumenta por ε , i.e., $\mu \leftarrow \mu + \varepsilon - \varphi$
operação cancelada (atraso, defeito, etc.)	recebe o valor φ do OpH, i.e., $\pi \leftarrow \pi + \varphi$	diminui o total dos créditos por φ , i.e., $\mu \leftarrow \mu - \varphi$

Adaptado de Leitão e Restivo (2006).

3.5 MECANISMOS PARA RECONFIGURAÇÃO

A supervisão da execução das estratégias de controle de um SRM também envolve definir alguns parâmetros relacionados ao tempo. Um exemplo é o “mecanismo de estimação do tempo de ação de reconfiguração” proposto por Porcarelli et al. (2004) relacionado ao instante e o intervalo de tempo necessários para definir quando ações de reconfigurações do sistema devem ser aplicadas. Este mecanismo é considerado na arquitetura proposta, isto é, os OpHs representam componentes autônomos que podem encapsular a representação de serviços oferecidos por recursos para fornecer suas funcionalidades. Desse modo, o horizonte de tempo associado ao plano de produção definido no PrH é despachado pelo TH para o OpH que representa o respectivo serviço. Considera-se uma “janela de tempo” em que os processos dentro desse intervalo estão encapsulados no OpH. A aplicação desse mecanismo considera os parâmetros:

- (i) τ_0 para representar o instante de tempo que uma falha ocorre, o sistema detecta sua

ocorrência e inicia uma reação de tratamento de falha/reconfiguração;

- (ii) $\Delta\tau_c$ para representar o intervalo de tempo necessário para conferir se uma ação de reconfiguração obteve sucesso. Além disso, este parâmetro determina uma frequência para execução de ações de verificação se a falha foi recuperada, re-estimação e reaplicação dos parâmetros de intervalo de tempo se a falha não foi recuperada como previsto;
- (iii) $\Delta\tau_e$ para representar o intervalo de tempo de estimação para recuperação do sistema. Neste intervalo as ordens alocadas ao recurso com falha são canceladas.

Para aplicar estes parâmetros, algumas premissas devem ser assumidas:

- (i) os intervalos de tempo são considerados determinísticos;
- (ii) $\Delta\tau_e$ é um intervalo de tempo que pode ser estimado a partir de dados históricos; e
- (iii) $\Delta\tau_c$ é menor do que $\Delta\tau_e$.

O projetista deve analisar na aplicação, quantas ações de reconfiguração podem ser aplicadas para cada ação de reação à determinada falha e o intervalo de tempo máximo de tempo de recuperação (restabelecimento) $\Delta\tau_r$, que pode ser baseado no intervalo de tempo despendido em histórico de tratamento de falhas anteriores/referenciais. Segundo ainda Porcarelli et al. (2004), uma possível maneira para calcular os parâmetros de tempo acima citados é assumir 50% de $\Delta\tau_r$ para $\Delta\tau_c$ e 90% para $\Delta\tau_e$.

Transcorrido $\Delta\tau_c$, se a falha não está recuperada, é necessário re-estimar os parâmetros de tempo e cancelar as estratégias que estão planejadas para o novo intervalo de tempo, que é obtido somando $\Delta\tau_c$ a $\Delta\tau_e$. Neste caso, o sistema aplica outra ação de reconfiguração e verifica novamente a recuperação do sistema depois de transcorrido $\Delta\tau_c$.

O OpH deve remover a ordem planejada que seria executada dentro do novo intervalo de tempo ($\Delta\tau_c + \Delta\tau_e$) de sua lista de tarefas local, enviando mensagens para os THs responsáveis por cada uma destas ordens. Durante o intervalo de tempo que o recurso está indisponível, o OpH somente recebe novos comandos se estes puderem ser executados fora do intervalo de tempo estimado para restabelecimento. Após a execução da ação de reconfiguração, o OpH armazena as informações relevantes sobre a falha, como o intervalo de tempo de restabelecimento e a solução utilizada para recuperá-la. O registro destas informações permite o aprimoramento de futuras reações às falhas. Na Figura 21 tem-se um diagrama que ilustra o mecanismo descrito.

A autonomia de cada holon é neste caso associado ao “mecanismo fator de autonomia” conforme proposto por Leitão e Restivo (2006), isto é, uma variável discreta $\{baixo, alto\}$. Inicialmente, o OpH tem um fator de autonomia $\{baixo\}$, que permite a sua coordenação pelo

Figura 21 - Estimação do tempo de reconfiguração

SuH, TH e PrH, executando quando disponível, os planos que estes holons indicam. Quando uma falha ocorre e esta é identificada, inicia-se o tratamento da falha. Este mecanismo adapta o comportamento dos holons na presença de falha. As entradas para esta função são:

- (i) o valor da variável do fator de autonomia α ;
- (ii) o intervalo de tempo de restabelecimento $\Delta\tau_r$ que é o tempo estimado necessário para a recuperação da falha; e
- (iii) o parâmetro do tipo de falha ρ que indica a ocorrência de uma determinada falha.

No caso de um fator de autonomia $\alpha = \{baixo\}$, a ocorrência de uma falha representada por $\rho = \{ativa\}$, gera a mudança do fator de autonomia para $\alpha = \{alto\}$ e a seleção de um comportamento adequado para o tratamento da falha. Depois de transcorrido $\Delta\tau_r$, se o fator de autonomia é $\alpha = \{alto\}$ e $\rho = \{ativa\}$, isto significa que o sistema ainda não foi recuperado e $\Delta\tau_r$ é re-estimado. Caso a falha e a situação causada pela mesma tenha sido resolvida, situação indicada por $\rho = \{inativa\}$, o holon pode retornar a estrutura anterior à ocorrência da falha, mudando o *fator de autonomia* para $\alpha = \{baixo\}$. Quando a falha é removida, ou seja, quando a sua causa e suas consequências são sanadas, cada OpH reduz novamente seu *fator de autonomia* e retorna a estrutura de controle hierárquica anterior à ocorrência de falha. A Figura 22 ilustra a evolução deste mecanismo em RdP L/T.

A ideia é que o mecanismo “chaveamento de controle” envolve dois modos de operação: o “modo estacionário” em que o sistema de controle é coordenado de forma hierárquica e os OpHs seguem uma coordenação preestabelecida, e o “modo transiente”, em que os OpHs assumem o controle. Desse modo, é possível obter um comportamento de hierarquia parcial dinâmica.

Figura 22 - Evolução do fator autonomia.

Adaptado de Leitão e Restivo (2006).

O “mecanismo de propagação de mensagens” proposto por Leitão e Restivo (2006) é adotado para assegurar a agilidade de resposta às falhas. Este mecanismo possibilita previsibilidade para reconfiguração do sequenciamento de tarefas em outros OpHs que podem ser afetados pela falha. Desse modo, no caso da reconfiguração o mecanismo pode, por exemplo, permitir prever ações de prevenção/mitigação de acidentes ou evitar o *deadlock* do sistema de manufatura. Entretanto, para tornar possível a devida reação à falha, o projetista deve avaliar a necessidade do desenvolvimento de holons (OpHs, THs e SuHs) que executem ações se os efeitos da falha de um OpH ultrapassa os limites do encapsulamento previsto na concepção deste holon.

A Figura 23 ilustra um esquema do mecanismo adaptado para esta proposta e de seu diagrama de sequência UML. O mecanismo envolve o envio de mensagens da ocorrência de uma falha do OpH ao SuH ao qual está subordinado. Quando recebe esta informação, o SuH inicia a propagação de mensagens que indica a necessidade de reorganização. Os holons associados recebem a mensagem propagando também esta necessidade aos holons vizinhos, até que uma ação de reconfiguração é iniciada. O OpH pode prever possíveis ações locais para recuperar fisicamente um recurso afetado por uma falha, baseado em soluções usadas em ocorrências anteriores, e passa esta informação para o holon operacional OpH que é responsável por executar a operação de reconfiguração.

Figura 23 - Propagação de mensagem de ocorrência de uma falha.

Adaptado de Leitão e Restivo (2006).

3.6 SÍNTESE DO CAPÍTULO

Este capítulo apresenta a arquitetura do controle de SRM. A solução de controle de um SRM deve integrar *hardware* e *software* em diferentes arquiteturas e especificações heterogêneas de diferentes tecnologias. Com adoção da arquitetura de controle proposta, é possível garantir colaboração de funcionalidades baseada em holons e usar as vantagens de plataformas bem definidas de sistemas multiagentes. Além disso, no nível de máquina e de dispositivo composto por *hardware* e *software*, CLPs, dispositivos de IHM I/O, controladores de robô, etc., os quais têm soluções proprietárias dos fabricantes de automação, é possível tratar a interoperabilidade, usando a linguagem XML, o que pode tanto garantir a implementação de sistemas heterogêneos como a atualização de sistemas legados. Existem poucos *hardwares* usados nas indústrias de manufatura que já sejam baseados em SOA e HMAS.

Assim, a arquitetura proposta garante características de autonomia, recursão e integração através da interação entre os holons. Nesta arquitetura, um holon pode pertencer a diferentes holarquias tanto do mesmo nível de controle como de níveis diferentes. No nível de controle dos dispositivos físico das máquinas, a geração do código de implementação segue a linguagem gráfica proposta na IEC61131 (JOHN; TIEGELKAMP, 2010) ou o código C. Nos níveis de controle mais altos, cada classe de holon é implementada por um agente na plataforma JADE+WADE, que permite interoperabilidade entre diferentes sistemas operacionais. De fato, JADE+WADE facilitam a implementação do controle de SRMs, pois é possível desenvolver códigos Java que representam os detalhes de um processo produtivo, baseado em interfaces

gráficas de fluxos, os quais podem ser editados, re-fatorados, depurados e podem incluir métodos, classes, referências para classes externas e assim por diante. Ou seja, JADE+WADE é adotada pois facilita a reconfiguração do sistema de controle de manufatura, e traz a abordagem baseada em *workflow* para o nível interno da lógica do sistema e, além disso, fornece mecanismos avançados de administração e tratamento de falhas. Por outro lado, JADE+WADE prevê a flexibilidade no caso do desenvolvedor não desejar utilizar gráficos, pois os padrões podem também ser escritos diretamente usando os padrões Java.

Assim, esta arquitetura de controle garante a integração dos diferentes níveis de controle de um SRM que é tratado de forma mais flexível do que a representação clássica e hierárquica de controle.

4 IMPLEMENTAÇÃO DA ARQUITETURA DE CONTROLE

Este capítulo descreve o método para a implementação da arquitetura de controle proposta no Capítulo 3. Para auxiliar o entendimento das soluções de controle propostas para sistema reconfigurável de manufatura, este capítulo apresenta, em conjunto com a descrição do método, alguns exemplos e modelos padrões de holons. Os modelos padrões servem como uma estrutura base para o projetista desenvolver o sistema de controle com módulos pré-existentis.

Por fim, mecanismos propostos para tratamento de falhas e tomada de decisão são apresentados. O método proposto está representado em PFS na Figura 24 sendo dividido em três etapas representadas por *atividades* para tratar a inerente complexidade da especificação de um controle de sistema reconfigurável de manufatura. Neste método:

- (i) cada uma das etapas é representada por um elemento *atividade* do PFS. Esta representação explora a vantagem do PFS quanto a definição de um protocolo de comunicação entre os envolvidos no projeto. Se necessário, está previsto o retorno às etapas anteriores para revisões das especificações. Para diferenciar o PFS usado na descrição do método das especificações também descritas em PFS, são acrescentados os símbolos correspondentes de cada holon da arquitetura no caso dos PFSs das especificações.
- (ii) na descrição de cada etapa tem-se a identificação do “ator” e dos “artefatos” gerados e utilizados¹². Por simplificação, na Figura 24 os atores e artefatos não são representados. Nas Figuras 25, 36 e 48, artefatos resultantes de fonte externas são destacados em linha tracejada verde e os artefatos gerados pela aplicação do método estão destacado em linha tracejada azul. Atores estão em linha tracejada amarela;
- (iii) o conceito de HMAS está encapsulado no detalhamento das etapas e no uso da plataforma de implementação de sistemas multiagentes;
- (iv) o conceito de SOA é considerado na definição do projeto, encapsulado na composição dos holons e na utilização de plataformas que são baseadas em agentes que requisitam serviços.

Neste trabalho se assume que a reconfiguração é utilizada para melhorar o desempenho do sistema para garantir a qualidade, ou inclusão de novos tipos de produto ou ainda aumento de

¹²Ator e artefatos dentro do contexto de desenvolvimento de um software baseado em SOA representam, respectivamente, um usuário (ou outro sistema) e o produto de uma ou mais atividades (FUGITA; HIRAMA, 2012).

Figura 24 - PFS do método de especificação e implementação do controle de SRM.

demanda em termos de produtividade. São adotados mecanismos que permitem autonomia e flexibilidade de reconfiguração (DA SILVA et al., 2014), seja para mudança de produção ou tolerância a falhas/reconfiguração do sistema. Desta forma, na lista de requisitos devem ser levantados os aspectos relacionados: a identificação das falhas, as especificações para o diagnóstico de falhas e as estratégias para tratamento de falhas e para reconfiguração.

Como citado anteriormente no Capítulo 2, as características de SRMs devem ser consideradas nos requisitos. Ou seja, o SRM deve considerar a “responsividade”, de modo que:

- (i) o sistema de manufatura e seus equipamentos devem ser projetados de forma que sua estrutura física pode ser facilmente modificada; e
- (ii) o sistema de manufatura e seus equipamentos devem ser projetados para produção e inspeção de uma certa família de produtos.

Além disso, os requisitos dos SRMs devem envolver: personalização (limitada a uma família de produtos), convertibilidade, escalabilidade, modularidade, integrabilidade, diagnosticabilidade, interoperabilidade e portabilidade. Por exemplo, faz-se necessário incluir no sistema, máquinas multifuncionais, algum tipo de redundância em equipamentos, rotas alternativas, possibilidade para mudar recursos/máquinas, etc. A Tabela 7 relaciona outros aspectos que devem ser levantados nos requisitos para identificar as estratégias de tratamento de falhas.

No processo de decisão, principalmente quando o diagnóstico não é preciso, considera-se a utilização de regras de inferência de sistemas especialistas baseado em raciocínio. Por exemplo, se o diagnóstico acusa que um sistema formado por dois componentes está apresentando algum defeito, o que não permite que o sistema cumpra com seu objetivo, pode-se inferir que o componente em falha é o que possui o *Mean Time Between Failures* (MTBF), e então adotar as ações necessárias para correção desta falha. Se isto não resolver o problema, o outro

Tabela 7 - Aspectos relacionados a requisitos de tolerância a falhas

aspecto	questões
ocorrência de falhas	que falhas podem ser detectadas e diagnosticadas?
	em que condições estas falhas podem ser detectadas e diagnosticadas?
	quais as soluções e o tempo necessário para recuperação destas falhas de acordo com dados existentes ou históricos?
	que falhas podem levar a uma reconfiguração do sistema?
componentes do sistema	que parâmetros podem ser reconfigurados?
sensoriamento	que informações devem ser coletadas e em que frequência?
arquitetura do sistema	que recursos podem ser utilizados para reconfigurar o sistema?
	qual o grau de redundância a ser considerado?
ação de reconfiguração	qual grau de degradação de desempenho é aceitável em cada processo?

componente é o que está com falha. Outra forma de implementar o processo de tomada de decisão é definir prioridades de execução entre as alternativas.

4.1 MODELAGEM CONCEITUAL

Esta etapa é desenvolvida pelos analistas de requisitos. O artefato utilizado é a lista de requisitos e/ou a lista de melhorias, a qual define especificações do sistema, tais como objetivo, objeto de controle, dispositivos de controle, subsistemas, características estruturais da planta física sob controle, *layout* e recursos. A Figura 25 ilustra o PFS desta etapa. A validação de requisitos envolve uma extensa área de pesquisa necessitando de especialistas que tenham amplo conhecimento do domínio de aplicação e com aspectos que não podem ser completamente automatizados. Considera-se que no método proposto a lista de requisitos/melhorias (neste caso, para atualização de sistemas legados) está previamente definida. Os artefatos gerados nesta etapa são os modelos em PFS e a especificação da arquitetura de controle.

4.1.1 Definição da Ontologia

A especificação do controle de um SRM é baseada, inicialmente, em definição de ontologias, para prover um modelo semântico adequado para identificação e das interações envolvidas. Cada domínio de aplicação tem sua padronização específica da semântica e eficiente interação. Nesta proposta, a especificação da ontologia é feita em uma abordagem *top-down* e usando a hierarquia de conhecimentos disponibilizada pelo PFS.

Assim, a especificação da ontologia define ordens, planos de produção, capacidade, sistema

Figura 25 - PFS da etapa modelagem conceitual.

de transporte e outros componentes. Esta especificação permite reconfigurar o sistema com mais facilidade para, por exemplo, inclusão de um novo tipo de produto, adicionar um novo plano de produção e suas novas operações. Por exemplo, quando um novo produto é adicionado, é suficiente estender a ontologia para introduzir o novo produto e seus parâmetros, adicionar um plano de produção para aquele produto, e para adicionar regras para negociação dos holons para o específico plano de produção.

Esta especificação inclui diferentes aspectos de sistemas de manufatura e seus processos de negócio, tais como especificação de ordem de cliente, definição de plano de produção, transporte e manipulação de material. A ontologia é definida usando o PFS e este trabalho não foca um estudo extenso sobre ontologias, as quais podem ser implementadas em técnicas específicas, tais como OWL e usando *software* como Protegé. Entretanto, o uso do PFS facilita o entendimento e serve como uma base para especificação de um controle de sistema reconfigurável de manufatura. Os projetistas devem entender esta ontologia e serem capazes de manipulá-las, pois é necessário para cada domínio de aplicação definir claramente suas especificidades. Além disso, os componentes do sistema devem adotar um padrão semântico desde a etapa de projeto. Neste caso, a vantagem desta abordagem é que incluindo subclasses de holons, o sistema de controle deve manipular as mudanças necessárias dos processos envolvidos.

A Figura 26 apresenta uma proposta da especificação da ontologia, a qual é baseada em estudos de ontologias previamente citados, tais como ANSI/ISA (1995), Obitko e Mařík (2003), Obitko et al. (2010). Trata-se de um tipo de modelo que deve ser modificado baseado no domínio, hierarquia dos processos produtivos e suas possíveis falhas. Nesta figura, processos de negócios representam diversos processos da empresa, onde, por simplificação, somente alguns modelos são refinados. Neste exemplo, uma ordem de cliente contém uma ou

mais ordens de produto para permitir a execução de certa quantidade do mesmo tipo de produto os quais devem ser entregues nos seus devidos prazos. O produto é especificado pelo nome e os parâmetros que eles devem ter. O plano de produção representa o modo que descreve como juntar produtos intermediários, sobre os quais são realizadas operações para se obter o produto final. Cada operação é caracterizada por uma breve descrição quanto a sua execução, contendo: o tipo de máquina, os recursos necessários, as precedências e o prazo estimado. O conjunto das operações formam um determinado serviço disponibilizado pelo recurso. Um processo de manufatura envolve a execução consecutiva de ordens de trabalho que têm uma programação de operações e/ou serviços, que por sua vez estão associadas com uma prioridade de execução, requisitam lista de materiais e recursos. Esses recursos são equipamentos que têm uma localização definida e podem ser afetados por uma falha. Estratégias de reconfiguração devem recuperar a funcionalidade do sistema quando este for afetado por uma falha. Assim, diferentes processos de uma ordem de cliente são representados, seguindo a ontologia definida nesta subetapa.

Figura 26 - PFS do modelo padrão para definição da ontologia.

4.1.2 Modelagem conceitual dos holons

Baseado no levantamento anterior e na especificação da ontologia, os holons são identificados. Um PrH é definido para representar cada processo que envolve a matéria prima, produto intermediário ou produto final do SRM. Cada PrH contém todo o conhecimento que

envolve a definição das funções de controle relacionados com cada produto a ser processado. Cada processo de PrH tem um fluxo que envolve a sequência de serviços que devem ser executados para obter o produto final e atender o plano de produção do sistema. O PrH considera as estratégias adotadas nos planos de produção de acordo com um fluxo de precedências e a combinação pré-definida entre PrHs.

A Figura 27 ilustra as atividades que representam os processos do modelo padrão de um PrH. As atividades do PrH atuam no plano de produção e a Figura 28 ilustra um exemplo da representação deste plano. O modelo indica os PrHs que representam os processos relacionados a matéria-prima (PrH-MP1), aos produtos intermediários (PrH-PI1 e PrH-PI2) e aos produtos finais (PrH-FP1 e PrH-FP2). Desta forma tem-se definidos dois planos de produção: $[\text{PrH-MP1}] \rightarrow [\text{PrH-PI1}] \rightarrow [\text{PrH-PF1}]$ e $[\text{PrH-MP1}] \rightarrow [\text{PrH-PI2}] \rightarrow [\text{PrH-PF2}]$. Neste exemplo, cada processo envolvendo um PrH tem um serviço associado que deve ser executado.

Figura 27 - Modelo conceitual padrão do holon de produto.

Figura 28 - Exemplo de modelagem dos processos envolvendo um plano de produção representado por PrHs.

O TH define estratégias de controle para execução de cada processo produtivo. Por exemplo, em cada PrH tem um fluxo interno de produção envolvendo um ou mais serviços e THs representam as estratégias de controle baseadas na colaboração de holons e nos requisitos dos produtos (por exemplo, tempos de execução e sequência das ordens) para atender o plano

de produção. Os THs coordenam a alocação e execução dos produtos/serviços de acordo com os OpHs. Para cada TH deve-se especificar tanto as tarefas de controle de operações normais como as tarefas sob a ocorrência de falhas, bem como as medidas a serem tomadas em casos de situações emergenciais, tais como falta de energia, erros de operação, etc. Estas especificações regulam o comportamento do sistema definindo, por exemplo, os mecanismos de tratamento de falhas; a alocação de recursos; a negociação com OpHs; a execução dos processos de negócios, suas ordem de trabalho e lista de operações; o gerenciamento da execução e sua supervisão; e o gerenciamento das estratégias de reconfiguração, tais como aquelas relacionados com o fator de autonomia, o chaveamento do modo de controle e a reorganização estrutural da arquitetura. Assim, o modelo conceitual padrão de THs descreve como a composição de holarquias e a reconfiguração devem ser realizadas. A Figura 29 ilustra este modelo.

Figura 29 - Modelo conceitual padrão do holon de tarefas.

Para modelar os THs também deve ser feito o detalhamento das ações e operações para execução das ordens de produção e para os casos de reconfiguração. Para modelar os THs é usado o PFS onde cada solução de controle é representada por uma atividade, a interação entre elas é representada por fluxos primários e a interação entre as estratégias por fluxos secundários. A Figura 30 apresenta um exemplo da modelagem de uma solução de controle para uma estação de trabalho (PWS) composta por quatro máquinas ("CNC A", "CNC B", "RMT acb", "RMT rcb"). Neste exemplo são especificados quatro soluções de controle que são executadas de acordo com a escolha de três tipos de produtos. Neste modelo, a flexibilidade do plano de produção de cada peça é considerada para permitir a reconfiguração do sistema. Por exemplo o produto "bcb" pode ser usinado tanto na "CNC A" como na "CNC B". O sistema de controle pode reconfigurar a rota da peça de acordo com o plano de produção e a disponibilidade das máquinas.

O SuH contém o conhecimento para a supervisão do produto/serviço executado pelos holons, coordenando suas operações, registrando as habilidades de cada componente e

Figura 30 - Exemplo de modelagem de serviço de uma estação de trabalho (PWS).

forneendo funções combinadas com outros holons do sistema de controle. Uma das funções do SuH envolve a coordenação entre os holons monitorando as operações que estão sendo executadas. Ou seja, o SuH coordena as funções relacionadas a interfaces (representadas por *distribuidores*) e a orquestração de estratégias dos holons. A Figura 31 ilustra essas funções representadas por *atividades* relacionadas entre outras funções, tais como o recebimento de anúncios (isto é, tratamento de mensagens recebidas de outros holons), supervisão das tarefas sob execução, tratamentos de falhas e, se necessário, a reconfiguração do sistema. Além disso, há a *atividade* de [orquestração] para as ações que formam uma holarquia otimizada de acordo com a funcionalidade dos holons. A orquestração também envolve a troca de mensagens entre holons, os quais são representados nos SuHs usando *distribuidores*. Assim, o SuH é concebido para orquestrar a composição de holarquias pesquisando a funcionalidade dos holons para alocar uma ordem de produção. Em caso de adição ou remoção de processos ou funcionalidades, ocorrência de falha e para otimizar a produção de novas demandas, o SuH aplica estratégias de reconfiguração. Além disso, para a monitoração do sistema, o SuH envia mensagens para holons específicos em intervalos pré-definidos para verificar as métricas do sistema.

O modelo conceitual padrão do OpH está representado no PFS da Figura 32. Inicialmente, o OpH inicializa seus componentes, registrando suas habilidades no programa de tarefas no SuH ao qual está subordinado, de acordo com a arquitetura de controle. Quando a ordem é finalizada, o OpH deve confirmar se houve sucesso ou não na execução.

Figura 31 - Modelagem conceitual padrão do holon supervisor.

Figura 32 - Modelo conceitual padrão do holon operacional.

O OpH também considera os recursos físicos da planta tais como, equipe de manutenção e operação, equipamentos e dispositivos de controle. O OpH estabelece o comportamento destes recursos de acordo com os objetivos e habilidades de cada um, e segue uma lista planejada de operações que o recurso tem de executar para entregar seu serviço. Observa-se que nem todos os recursos disponíveis são candidatos a estarem associados a um holon, pois, analisando as dependências físicas entre os objetos de controle, é possível agregar um ou mais recursos em um único holon. Para a modelagem do OpH devem ser analisados todos os estados normais e os estados de falha, bem como as possíveis condições que podem levar o recurso de um estado

a outro. Estes estados são relacionados entre si, conforme ilustra a Figura 33 e, portanto, devem ser considerados durante cada aplicação. Para a representação de humanos, o modelo é similar a esta representação, sendo que neste caso dois estados são representados: disponível e indisponível. Esta representação do OpH torna possível o reuso de modelos, a construção de modelos de sistemas supervisórios, e a interação humana nos processos de um SRM como processos interativos que ocorrem com gerente, venda, contabilidade, produção, garantia de qualidade, de embalagem, transporte, etc.

Figura 33 - Exemplo do PFS do objeto de controle representado pelo holon operacional.

4.1.3 Orquestração dos modelos conceituais

Nesta subetapa, os processos interativos (orquestração) tais como solicitação de produtos, execução de serviços, tratamento de falhas e reconfiguração são especificados com base em modelos PFS. Em cada domínio de aplicação, interações adicionais entre os holons precisam ser definidas as quais envolvem especificidades da aplicação. A orquestração é inicialmente especificada considerando as possíveis interações entre os holons. Isto é, representa-se o fluxo principal e o fluxo secundário de itens/informações entre as atividades.

Neste contexto, são descritos dois processos interativos entre holons, representando a colaboração para execução de uma ordem de cliente e a reconfiguração do sistema. Estas interações bem como de outros processos são detalhados na etapa de modelagem funcional.

A Figura 34 ilustra o modelo da orquestração para “execução da ordem de um cliente” em que as atividades consideradas são aquelas definidas nos modelos individuais dos holons apresentados na Subseção 4.1.2. Por simplificação, somente estão ilustradas nesta figura as atividades que têm algum tipo de processo interativo. Neste exemplo, o PrH envia o plano de produção ao TH que deve iniciar uma negociação com os OpHs para alocação das ordens de trabalho. Os OpHs, por sua vez, devem responder ao TH quanto ao término da

execução das operações e esta informação é também enviada ao PrH. O SuH é responsável pela orquestração através das suas atividades de supervisão. Os produtos possuem, cada um, seu plano de produção que deve ser realizado. Ou seja, o plano de processo é definido para o produto, tendo em consideração suas variáveis (por exemplo, tipo de montagem) e parâmetros de operação (por exemplo, espessura do processo de soldadura tipo de usinagem). Cada PrH contém uma sequência de serviços que devem ser cumpridos e interage com os holons, a fim de garantir a realização do produto de acordo com o plano de processo.

Figura 34 - PFS que representa a orquestração para execução de uma ordem de trabalho.

Quando uma estratégia de reconfiguração é executada, o SuH define uma prioridade de execução e comunica aos demais holons para assegurar maior agilidade de reação. Cabe ressaltar que algum grau de redundância deve ser considerado para que algumas funções de dispositivos em falha possam ser executadas por outros dispositivos, tais como adoção de recursos multifuncionais.

A Figura 35 ilustra o modelo da orquestração para reconfiguração em que as atividades definidas nos modelos individuais dos holons são relacionadas. Ao receber do diagnosticador de um OpH que ocorreu algum tipo de falha, o SuH divulga esta informação a outros holons e, em função disso, o TH inicia um processo de revisão de ordens. Em paralelo, ações de reconfiguração são executadas, as quais são baseadas no mecanismo “decisor”.

4.2 MODELAGEM FUNCIONAL

O papel desta etapa é do analista de processos, onde os artefatos utilizados são os modelos em PFS; os artefatos gerados são os modelos em IOPT dos holons e os dados de simulação/análise desses modelos. A Figura 36 ilustra o PFS desta etapa que é subdividida em modelagem dos holons em IOPT, orquestração dos processos, modelagem de mecanismos

Figura 35 - PFS que representa a orquestração para interação em caso de reconfiguração.

de diagnóstico e tomada de decisão, análise e revisão dos modelos. Se na análise dos modelos é verificada a necessidade de mudanças/melhorias, retorna-se à primeira subetapa.

Inicialmente, cada modelo PFS é refinado utilizando IOPT. Em seguida, são incluídos os intertravamentos (representados por expressões *guards*) de cada transição, os comandos de saída e de entrada de cada lugar (*output and input action*) e os eventos de entrada e de saída (*input and output events*). Por último, o distribuidor, que no modelo em PFS é também usado na orquestração entre os holons, são representados por *asynchronous channel*, lugares auxiliares ou expressões lógicas definidas por *guards* em IOPT. Assim, a modelagem funcional em IOPT é usada para especificar o fluxo interno (*coreografia*) e o detalhamento em IOPT da orquestração para colaboração entre os holons.

Alguns modelos em PFS dos holons de níveis superiores (PrHs e alguns THs/SuHs) podem com relativa facilidade serem transformados para os modelos JADE+WADE. Entretanto, mesmo para esses holons, seus modelos são também desenvolvidos em IOPT, para a análise da dinâmica do fluxo interno dos holons. Portanto, todos os processos de negócio são modelados usando IOPT para efeito de análise. Esses processos são assíncronos, porém executados de forma síncrona e em conjunto, e portanto requerem troca de mensagens considerando esta natureza. Neste caso, esta sincronia é registrada pelos SuHs e representada por *Asynchronous Channels (AC)*.

Figura 36 - PFS da etapa modelagem da funcional.

4.2.1 Modelagem dos holons em IOPT

Na etapa de modelagem conceitual, a estrutura de cada holon é representado em PFS por uma atividade (todos os holons) ou por um distribuidor (alguns casos de SuHs). Nesta subetapa, cada modelo PFS é refinado utilizando a IOPT. Em seguida, são inclusos os intertravamentos, guard, eventos de entrada/saída, de cada transição e lugar.

No modelo funcional em IOPT, para facilitar o entendimento, uma padronização é adotada na definição dos lugares e transições, considerando-se:

- (i) um “caminho (fluxo) principal”, onde lugares são representados pela sequência $p_1, p_2, \dots, p_n$ e transições por $t_1, t_2, \dots, t_n$, sendo $(n \in \mathbb{Z})$;
- (ii) um “caminho (fluxo) secundário”, onde:
 - (a) uma transição oriunda de um lugar m é nomeado por $p_m/t_1, p_m/t_2, \dots, p_m/t_n$ e um lugar nesta sequência é nomeado por $p_m/p_1, p_m/p_2, \dots, p_m/p_n$, sendo $(m \text{ e } n \in \mathbb{Z})$. Ou seja, é incluída uma “/” para indicar que a sequência é oriunda de um lugar, representando um processo alternativo ao principal;
 - (b) um lugar oriundo de uma transição m é nomeado por $t_m.p_1, t_m.p_2, \dots, t_m.p_n$ e uma transição nesta sequência é nomeado por $t_m.t_1, t_m.t_2, \dots, t_m.t_n$, sendo $(m \text{ e } n \in \mathbb{Z})$. Ou seja, é incluído um “.” para indicar que a sequência é oriunda de uma transição, representando um processo paralelo ao principal.

O modelo funcional do PrH é mostrado na Figura 37. A padronização da descrição segue o citado no parágrafo anterior. Por exemplo, a atividade [elabora planos de produção] inicia

na transição t_3 e no processo em paralelo que inicia no lugar " $t_3.p_1$: verificando dados" e finaliza no lugar " $t_3.p_4$ ". Para melhor entendimento, alguns modelos funcionais têm as atividades do respectivo modelo conceitual dos holons indicadas e delimitadas em linhas de cor cinza.

A modelagem de PrHs envolve o fluxo interno de processos referente a cada produto, que envolve a execução de uma ou mais operações gerando outros produtos que podem ser intermediários ou final. Um exemplo da representação em IOPT de processos de um produto por um PrH está ilustrada na Figura 38. Todos os PrHs têm um lugar que representa seu estado inicial e a sequência de transições e lugares os quais representam requisições de serviços e a execução do mesmo, respectivamente. No caso de um PrH representar uma matéria-prima, ele deve ter um lugar que representa um possível estado inicial do plano de produção com a respectiva marcação inicial (neste exemplo: "inicial_MP"). Se for um produto intermediário, o modelo do PrH inicia e finaliza com lugares que representam uma interface que deve ser unida com outros PrHs (neste exemplo, respectivamente: "inicial_PI" e "interface_PI"). Se o PrH representa um produto final, o modelo tem um lugar que representa o estado final do plano de produção (neste exemplo: "interface_PF"). Entre esses lugares acima citados, o modelo IOPT define a sequência de serviços que deve ser requisitado para obtenção de cada PrH. A Figura 39 apresenta o detalhamento do modelo funcional padrão do TH em IOPT.

O modelo do holon de tarefas envolve a definição das estratégias de controle. A Figura 40 mostra um exemplo de estratégia de controle para um TH-[manufatura]. Neste exemplo, o TH aguarda a composição de um plano de produção para iniciar a execução.

O modelo padrão das funcionalidades dinâmica do SuH está representado na Figura 41. Neste modelo está descrita a interação das atividades relacionadas às funções de supervisão da composição de holarquias, monitoração, supervisão do tratamento de falhas e reconfiguração baseada na orquestração e funcionalidade dos holons.

O modelo padrão das funcionalidades do OpH está representado na Figura 42. O modelo detalha as atividades do seu modelo conceitual, que envolve alocação de recursos para execução de ordens. Além disso, no OpH também são representadas ações relacionadas a monitoração de processos e reconfiguração. Cada objeto de controle inclui a representação dos seus estados com comportamento normal e com falhas, além das condições para mudanças entre eles. O atendimento dessas condições depende da lógica representada por elementos da IOPT. Assim, os modelos de OpHs consideram o comportamento real do SRM incluindo a influência do ambiente e das falhas. Para representar cada um dos objetos de controle em IOPT considera-se o seu possível conjunto de estados e as condições necessárias para as mudanças de estados. Para modelá-los com o requisito de tolerância a falhas considera-

Figura 38 - Exemplo de IOPT da representação de produtos pelos PrHs.

do ambiente da solução de controle também deve ser modelada considerando os estados (normal e sob falhas) dos dispositivos de atuação em conjunto com os estados (disponível ou indisponível) dos outros dispositivos de controle. Lugares representam estados do objeto e transições a mudança para um outro estado. Esta mudança pode ser oriunda de um sinal de comando para um outro estado normal (sem falhas) ou de algum evento externo que causa um estado de falha. A Figura 43 ilustra esta modelagem, onde os lugares da IOPT representam os estados normais e a possível falha de um objeto de controle, neste caso, de um atuador solenoide denominado [1Y1]. No modelo é possível identificar dois estados normais de [1Y1], que são o estado onde o atuador está em posição avançado “1Y1OffPistonAdvanc” ou em posição recuado “1Y1OnPistonRetreat”, e um estado que representa uma situação falha “1Y1OffPistonAdvFault”, onde o atuador está em posição emperrado em uma das posições anteriores. Conectores sincronizam este modelo com outros modelos relacionados e com a estratégia de controle adotada. Por exemplo, os guards “G:1Y1=0” e “G:1Y1=1” representam o recebimento de comando do solenoide [1Y1] para enviar sinais para avançar o êmbolo do pistão [1A], ao qual este [1Y1] está conectado. Enquanto que a output “1Y1=0” representa um sinal de comando de saída para o solenoide. Existem dois input events: “1Y1SFault” que indica uma situação de mudança de um estado normal para o estado de falha; e “HardwareRepair” que a finalização do reparo, isto é, a correção da falha pela manutenção.

Figura 39 - IOPT do modelo padrão das funcionalidades do holon de tarefas.

Figura 40 - Exemplo de modelo IOPT da funcionalidade do holon de tarefas de acordo com uma estratégia de controle.

4.2.2 Orquestração dos modelos funcionais

A composição dos modelos representa a troca de mensagens e a sincronização dos holons. Os modelos dos holons são compostos para formar holarquias usando elementos da IOPT (aqui denominados conectores). Por exemplo, um conector do tipo ação (`input` ou `output action`) pode ser relacionado a um lugar. Os intertravamentos são considerados nas transições associando as condições para mudança de estado por meio de conectores que podem ser `input event`, `output event`, `guard expression` e `asynchronous channels`. A composição entre os holons pode também ser feita por meio de lugares auxiliares, sendo o principal benefício desta representação a visualização do disparo de marcas na sincronização dos processos. Conectores são definidos para sincronizar com outros processos relacionados, e com a estratégia de controle desejada e são definidos nos SuHs que desse modo endereçam interfaces entre os componentes para permitir a reconfiguração para substituição ou adição de componentes. Embora a utilização de lugares auxiliares aumente o número de elementos dos modelos em rede de Petri, por outro lado, isto facilita a visualização do envio e recebimento de sinais de controle, o que por sua vez facilita a tradução dos modelos para programas em linguagens de controladores industriais. Há ainda a necessidade de composição dos modelos de holarquias através da interação entre os modelos dos holons. Portanto, o projetista deve analisar a melhor forma de especificar novas interações.

A Figura 44 apresenta um exemplo da orquestração feita através de `Asynchronous Channels`. Neste exemplo estão detalhados em IOPT três THs: [manufatura], [inspeção da qualidade] e [embalagem]. Por simplificação, outros THs foram apenas indicados ([transporte], [ordem de vendedor] e [suprimento]). Esses THs colaboram entre si e ACs demonstram em quais transições é feita a troca de mensagens para informar certas situações.

Visto que ACs são elementos da IOPT usados para sistemas tipo Globally-Asynchronous Locally-Synchronous, na IOPT *web tool* deve-se definir quais modelos são localmente síncronos

Figura 42 - IOPT do modelo padrão das funcionalidades do holon operacional.

Figura 43 - Esquema de modelo IOPT de um recurso industrial representado por holons

do TH-[manufatura] foram especificados com o “td:4”. A vantagem de usar ACs é que a IOPT *web tool* permite decompor o modelo que contém todos os holons em arquivos contendo os submodelos que representam cada um dos holons, bastando dar um clique na aba “Decompose GALS” da IOPT *web tool*. A divisão é feita através da definição dos ACs. A Figura 40 mostra o submodelo TH-[manufatura] após ter sido usado a ferramenta para decompor GALS, onde é possível observar que *input* and *output* events fazem a função dos ACs. Por exemplo, o “event253” corresponde ao AC “RequerSuprimento” da Figura 44.

A Figure 45 apresenta outros exemplos tanto da modelagem conceitual quanto da funcionalidade e orquestração. Neste exemplo a orquestração é feita usando *input* event, *output* event, *guard* expression e lugares auxiliares. O exemplo tem a representação de um TH de processos de negócio (TH-[Processo de negócio]), que tem dois planos de produção relacionados. O PrH-[Plano de produção 1] utiliza o serviço de montagem (AMS) ou de usinagem (CNC), representados por OpH-[AMS WS] e OpH-[CNC WS], respectivamente. O OpH-[AMS WS] requisita a estratégia de controle representado pelo TH-[AMS A10] para fornecer seu serviço. Este TH gerencia uma operação cíclica de avanço ou recuo de um pistão elétrico, representado por OpH-[Atuador], onde um OpH-[sensor] está conectado a este pistão para informação sobre o movimento do mesmo e os três modelos em IOPT são sincronizados. Por exemplo, para avançar o pistão, TH-[AMS A10] envia o *output* signal “Advance = 1” para o OpH-[Atuador] habilitando sua *transição* com o “G:Advance=1”. Após seu disparo, o lugar “Piston Advanced” em OpH-[Atuador] envia um sinal através

Figura 44 - Exemplo de PFS da orquestração entre holons feito através de asynchronous channels.

do “G:Piston Advanced=1” para o OpH-[Sensor] por uma marca no lugar “Sensor On”, o qual por sua vez envia um sinal através de “G:Sensor On=1” para disparar a transição “Retreat” em TH-[AMS 10]. Esta transição tem um output signal “Retreat=1” para o OpH-[Atuador] recuar o pistão. Desse modo, o ciclo é re-inicializado e o movimento de avanço/recuo do pistão continua até a ocorrência ou de um input event que habilita a transição “Stop” em TH-[AMS 10] a qual tem o “G:Stop=1” relacionado a este evento, ou de um evento “Fault” que habilita um evento não observável em OpH-[Atuador], levando este objeto de controle ao estado representado pelo lugar “failure”. No caso do problema ser resolvido, o input event “Repair” retorna o pistão a posição inicial (recuado).

4.2.3 Modelagem de mecanismos de diagnóstico e tomada de decisão

Dois importantes mecanismos de controle devem ser modelados para completar os requisitos de diagnóstico de falhas e tomada de decisão. Para modelar o “diagnosticador”, é necessário definir eventos observáveis ou não-observáveis (SAMPATH et al., 1996) que são relacionados à

Figura 45 - Exemplo de PFS/IOPT que ilustra o desenvolvimento da modelagem conceitual e dinâmica.

especificação. Aspectos que permitem a construção desses mecanismos inicia-se na modelagem dos holons. Por exemplo, esses aspectos estão presentes nos componentes do objeto de controle, construção de modelos de estratégias de controle em PFS e IOPT, na definição de “eventos observáveis”, aqueles relacionados a comando de estratégia de controle do TH, e “eventos não observáveis”, que podem estar relacionados a falhas. Baseado nos modelos acima, nesta subetapa, o diagnosticador é desenvolvido de acordo com os seguintes passos:

- (i) representar um `lugar` para cada envio de comando referente à estratégia de controle nos objetos, inicialmente considerando suas operações em condições normais;
- (ii) relacionar as estratégias implementadas por meio de `transições` e `guards` tanto para os eventos observáveis como para os não-observáveis os quais podem ocorrer a partir do estado inicial do “diagnosticador”; e
- (iii) relacionar os estados que podem ser obtidos com os sinais de entrada ou saída (*input* ou *output signals*) de atuadores e sensores.

Se o diagnosticador não identifica o estado correto, então as causas devem ser inferidas para resolver possíveis conflitos. O mecanismo de decisão é chamado de “*decisor*” e suas regras de tomada de decisão podem ser baseadas em dados probabilísticos, relações lógicas ou outros formalismos, para inferir qual o estado real do sistema. Para a fase de decisão define-se regras do tipo “Se... Então...”. Na Figura 46 tem-se o exemplo de modelos do diagnosticador e do decisor para a estratégia e objetos de controle ilustrados anteriormente na Figura 45 e que também são replicados nesta figura para um melhor entendimento. Observa-se que o modelo do diagnosticador está sincronizado com a estratégia de controle através de `guards` (por exemplo, “G:Advance = 1” está relacionado a `output action` no recuado). Por sua vez, os modelos do diagnosticador estão relacionados aos modelos do decisor para cada execução da estratégia de controle: avançar ou recuar. Ao ser verificado a ocorrência de falha, os modelos do decisor, que também estão sincronizados com o modelo dos sensores, decide que, neste caso, executar reparos no pistão é o mais adequado.

4.2.4 Análise e Revisão dos Modelos

Esta subetapa é desenvolvida com apoio de ferramentas de edição de modelos, análise estrutural e funcional destes modelos. A análise é desenvolvida com base nas propriedades das redes de Petri e usando técnicas simulação implementadas em ferramentas para IOPT e de HMAS, especificamente a IOPT *web tool* e o *software* NetLogo. Este tipo de análise permite a reengenharia do sistema de controle durante a fase de projeto e a verificação das propriedades estruturais e comportamentais dos modelos em RdP, para derivar conclusões sobre o funcionamento do sistema, tais como:

- (i) *vivacidade*, que está relacionado com a ausência completa de bloqueios no sistema reconfigurável de manufatura;
- (ii) *acessibilidade*, para estudar as propriedades dinâmicas do sistema reconfigurável de manufatura;

Figura 46 - Exemplo de IOPT da modelagem dos mecanismos diagnosticador e decisor.

- (iii) reversibilidade, para verificar a recuperação do sistema após a ocorrência de eventos de interrupção da operação;
- (iv) conservação e limitação, para verificar a variação do número de itens em processos e em situações específicas.

Os modelos são assim editados e simulados na IOPT *web tool* (GOMES et al., 2007), onde as análises das propriedades de RdP são usadas para verificar os dados obtidos.

A validação do desempenho geral do sistema é feita usando NetLogo. A Figura 47 apresenta a tela de trabalho deste *software*. O Netlogo é bastante flexível e pode ser configurado para cada aplicação. Para melhor entendimento do processo de transformação de modelos IOPT para a validação destes no Netlogo um exemplo de aplicação é descrito no próximo capítulo.

Figura 47 - Exemplo de modelagem de um sistema de manufatura usando NetLogo.

4.3 IMPLEMENTAÇÃO

Considera-se aqui que o desenvolvedor realiza a implementação do sistema de controle usando os artefatos modelos em rede de Petri. Os artefatos gerados são as especificações de programas de controle. A Figura 48 mostra o PFS esta etapa.

A implementação é realizada baseada nos modelos de IOPT e padrões FIPA disponíveis na plataforma JADE+WADE. Desse modo, é possível gerar as especificações de programas do sistema de controle. Para implementação, os modelos resultantes são interpretados como

Figura 48 - PFS da etapa implementação.

especificações de programas de controle a serem executados por controladores programáveis (nível de controle local) e computadores (nível de controle supervísório). Essa etapa também compreende a codificação, parametrização e desenvolvimento de interfaces específicas de cada equipamento. No nível de controle de máquinas, devido suas similaridades com RdP, os modelos são transformados na linguagem SFC ou em programas em linguagem C utilizando a *IOPT web tool* (GOMES et al., 2007). Nos dispositivos com mais recursos de processamento de informações, utiliza-se JADE+WADE, devido suas características para implementação de sistema multiagente e de interfaces gráficas para desenvolver o programa em Java usando *workflow*. Além disso, JADE+WADE permite desenvolver agentes que invocam WebSs.

4.4 SÍNTESE DO CAPÍTULO

Com a aplicação do método proposto e seus procedimentos, é possível a partir das especificações dos requisitos do sistema reconfigurável de manufatura obter os componentes da sua arquitetura de controle (Capítulo 3) bem como as linhas de código do controle de SRM utilizando a combinação de técnicas de SOA, HMAS e RdP.

O método facilita alterações no sistema para realizar sua manutenção ou reconfiguração. Além disso, foi considerada certa flexibilidade no próprio método, ou seja, tem-se representação modularizada por modelos de produtos, processos ou dispositivos.

Com este trabalho tem-se uma solução prática, para implementação do controle de sistemas reconfiguráveis de manufatura, com a obtenção de programas em linguagem gráfica baseada na norma IEC61131 (JOHN; TIEGELKAMP, 2010), especialmente, o SFC devido suas similaridades aos modelos em RdP, ou usando o código C (através da ferramenta *IOPT web*

tool para geração automática).

A abordagem combinada do PFS com a IOPT permite modelar a estrutura do sistema bem como o seu comportamento em uma maneira mais simplificada e transparente do que o uso de representações baseadas em RdP ordinárias. Utiliza-se sub-rotinas encapsuladas em atividades do PFS, o que permite reuso dos modelos. Estas operações (*atividades*) podem ser executadas quantas vezes forem necessárias. Além disso, usa-se a decomposição funcional, ou seja, um problema é dividido em problemas menores (*sub-atividades*) sucessivamente, que devidamente combinados formam os componentes do sistema de controle.

Os holons são divididos em classes onde as sub-rotinas e os dados estão em uma única estrutura. Os dados são protegidos através das sub-rotinas, e os dados também são objetos na modelagem. Na abordagem, o foco está em atribuir responsabilidades (*comportamento*) coesas e para cumpri-las, os holons trocam mensagens entre si.

Para garantir reuso ou alterações nas soluções de controle, o método utiliza conceitos de SOA para evitar um acoplamento forte entre componentes, facilitando a manutenção e reconfiguração/alteração no código dos programas de controle. Holons representam serviços e suas interfaces são definidas diretamente nos processos de negócio. Ele é a representação lógica de uma tarefa de negócio repetitiva que tem um resultado específico (por exemplo, verificação de ordens dos clientes e supervisionar a execução da manufatura). O tratamento de diferentes níveis de granularidade dos serviços possibilita que todas as operações consideradas podem ser identificadas nos processos de negócio em PFS/IOPT. O método também considera a definição de uma ontologia que facilita a comunicação entre equipes envolvidas no sistema de manufatura, bem como de clientes, os quais se tornam parte do processo.

Em relação às operações produtivas, a supervisão é realizada sincronizando os processos modelados em IOPT por meio de sinais de sensores do estado dos dispositivos, permitindo a geração de relatórios e gráficos de controle, a fim identificar meios para o aprimoramento do sistema. A introdução ou remoção de novos componentes pode ser descrita simplesmente com a adição ou remoção de novas *marcas* nos modelos correspondentes de IOPT, ou seja, a modificação dos modelos em IOPT dos holons. A GUI da plataforma JADE+WADE é utilizada para monitorar e controlar o estado dos agentes.

Dependendo do usuário envolvido na operação do sistema, as funções e interfaces podem ser revisadas. Por exemplo, um administrador pode ser responsável pelo cadastro de usuários, um gerente de utilidades ser responsável pela atualização de OpHs, um gerente de produção ser responsável pela criação de TH e re-inicialização do sistema, e um gerente de fábrica ser responsável pela criação de PrHs.

A comparação das condições e ações que precisam ser atendidas em cada tarefa de controle com as *transições* e *lugares* dos modelos em IOPT facilitam uma sistematização do

processo de localização de falhas. Desse modo, se o sistema de controle não resolver uma situação de falha, requisita-se a intervenção de um especialista que pode adotar outros métodos para localizar onde ocorreu essa falha.

Portanto, o método define meios de integrar todas as funções de controle, isto é, reduzir a distância entre o problema no mundo real e o modelo abstrato além de considerar a inclusão dos requisitos de reconfiguração de sistemas de manufatura desde a concepção até a sua implementação.

5 EXEMPLO DE APLICAÇÃO

O método descrito na seção anterior é aplicado em um sistema de manufatura denominado *Modular Production System* (MPS) (MPS, 2015) (Figura 49). O MPS é composto por estações de trabalho – *WorkStation* (WS) – montadas em bancadas com rodízios para permitir criar diferentes cenários de produção. Para esta aplicação, um levantamento físico é realizado em algumas das WSs que estão disponíveis no laboratório de sistemas e automação da Escola Politécnica da Universidade de São Paulo.

Figura 49 - Sistema modular de produção e diagrama da tela de trabalho do *software* NetLogo.

Cada estação tem seu respectivo controlador de modo que cada uma pode funcionar de modo independente como se fosse uma planta industrial autônoma. Os controladores podem ser conectados a um microcomputador por meio de um *hardware* dedicado. Cada microcomputador, por sua vez, está conectado à internet como se fosse uma conexão de uma fábrica independente. Desse modo, esta estrutura permite emular o caso onde os operadores também estão fisicamente distribuídos e atuando no sistema via internet, assim como os clientes que encomendam os pedidos e/ou que desejam monitorá-los. Ou seja, desse modo o MPS dispõe de facilidades para implementar mecanismos e operar como um sistema autônomo para avaliar um controle de sistema reconfigurável de manufatura.

Um levantamento foi realizado e descreve-se a seguir aspectos de cada estação, tais como a sequência de operações e os dispositivos de atuação, detecção, monitoração e controle:

- (i) Estação de alimentação – *Distributing WorkStation* (DWS): quando um produto é requisitado, esta estação fornece a peça “base cilíndrica” (ilustrada na parte de baixo e no centro da Figura 49). Estas peças estão em um *buffer* com capacidade para armazenamento de até oito bases cilíndricas e que tem um sensor capacitivo para detectar a presença da base cilíndrica que está na posição mais inferior. A base cilíndrica pode ser preta ($[bcb]$), vermelha ($[rcb]$) ou prata ($[acb]$). Esta última possui um revestimento metálico. A sequência de alimentação do tipo de base cilíndrica é aleatória, ou seja, a DWS não é capaz de identificar a cor nem as dimensões das bases cilíndricas. Quando a DWS recebe uma requisição, um pistão pneumático é acionado para retirar a base cilíndrica do *buffer*. Em seguida, um braço pneumático com uma ventosa, que funciona por sucção, remove a base cilíndrica desta estação e a transporta para a estação vizinha.

- (ii) Estação de teste e estação de inspeção - *Testing WorkStation* (TeWS) e *Inspection WorkStation* (IWS): estas estações controlam a qualidade e identificação das características físicas da base cilíndrica. A TeWS/IWS recebem uma base cilíndrica por vez, identificam a cor e realizam o controle dimensional da altura da base cilíndrica. A inspeção de cor é feita utilizando a informação de 3 sensores: um capacitivo, um indutivo e um óptico. O sensor capacitivo identifica a presença de uma base cilíndrica. O sensor indutivo identifica se uma base cilíndrica tem revestimento metálico (prata). O sensor óptico identifica a cor da base cilíndrica, diferenciando as bases cilíndricas de cor vermelha ou preta. Essas medições determinam se a base cilíndrica atende aos requisitos do pedido. Caso não sejam atendidos, a base cilíndrica é empurrada por um pistão pneumático para um *buffer* de descarte na SWS, e caso estejam atendidos, a base cilíndrica é colocada em um *pallet* que a leva para outra estação. A TeWS/IWS possuem ainda uma plataforma móvel com sensor piezoelétrico capaz de medir a altura da base. Os atuadores desta estação são o pistão pneumático e a plataforma móvel.

- (iii) Estação de montagem - *Assembling WorkStation* (AWS): esta estação é formada por duas subestações que em conjunto realizam a montagem das peças para obtenção do produto final. Observa-se que nesta estação, tem-se um “manipulador” que é um braço robótico com três juntas prismáticas acionadas por motor elétrico e com uma garra pneumática. Conforme ilustrado na parte de cima e no centro da Figura 49, nesta estação também existe um *buffer* para a peça “pino de cor prata”, um *buffer* para a peça “pino de cor preta”, um *buffer* para a peça “mola” e um *buffer* para a peça “tampa”. A AWS realiza a montagem unindo a base (nesta ordem) a um pino, uma mola e uma tampa. Tem-se um sensor capacitivo na posição de entrada da estação para detectar a presença de um *pallet* que contém uma base. O manipulador inicialmente retira a base cilíndrica e a coloca em

um local com um prendedor para impedir qualquer movimentação da base. Um atuador pneumático rotativo remove o pino escolhido do *buffer* e posiciona-o para retirada pelo manipulador que retira o pino e o coloca dentro da base. No *buffer* de mola existe um pistão pneumático que a posiciona para ser retirada pelo manipulador e colocá-la dentro da base sobre o pino. As tampas também são posicionadas no seu respectivo *buffer* por um pistão pneumático e o manipulador pega a tampa e a coloca de forma alinhada com a base, o pino e com a mola. A garra do manipulador então gira a tampa para fixá-la na base e completar a montagem do produto final que então é posicionado sobre o *pallet* para retirá-lo do subsistema de montagem. Os atuadores são três motores que acionam o manipulador, a válvula pneumática de abertura e fechamento da garra, e os três pistões pneumáticos que removem a mola, a tampa e os pinos de seus respectivos *buffers*. Assim, o objetivo da AWS é a montagem de produtos compostos pelas peças: uma “base cilíndrica” (preta ($[bcb]$), vermelha ($[rcb]$) ou prata ($[acb]$)), um “pino” (preto ($[bp]$) ou prata ($[ap]$)), uma “mola” ($[s]$) e uma “tampa” ($[co]$). Desse modo, na AWS pode-se obter os seguintes produtos:

- (a) “base preta + pino prata + mola + tampa” que é descrito como $[bcb+ap+s+co]$;
 - (b) “base preta + pino preto + mola + tampa” que é descrito como $[bcb+bp+s+co]$;
 - (c) “base prata + pino preto + mola + tampa” que é descrito como $[acb+bp+s+co]$;
 - (d) “base prata + pino prata + mola + tampa” que é descrito como $[acb+ap+s+co]$;
 - (e) “base vermelha + pino prata + mola + tampa” que é descrito como $[rcb+ap+s+co]$;
 - e
 - (f) “base vermelha + pino preto + mola + tampa” que é descrito como $[rcb+bp+s+co]$.
- (iv) Estação de transporte – *Transporting WorkStation* (TrWS): executa o transporte dos *pallets* com as bases para as estações e dos *pallets* para a retirada dos produtos finais. A TrWS é composta por um conjunto de esteiras e *pallets*. A parada do *pallet* em certo local é feita por pistões elétricos colocados nos pontos de parada, os quais acionam uma trava mecânica que barram fisicamente os *pallets* de acordo com sua posição. O *pallet* de armazenamento é removível e possui um espaço com dimensões compatíveis com a base cilíndrica ou produto final. O transporte é feito entre as estações e outros sistemas para a fabricação do produto final, retirada deste e armazenagem de *pallets*. Há sensores capacitivos que indicam a presença dos *pallets* em cada estação. Os atuadores são pistões elétricos que param os *pallets*. Pode haver filas de *pallets* esperando para entrar na AWS. Após a montagem, o produto final segue em um *pallet* até um *buffer*, em seguida o *pallet* é então liberado para outros serviços.

- (v) Estação de processamento – *Processing WorkStation* (PWS): nesta estação são emuladas as operações de usinagem/furação nas bases. Este exemplo de aplicação considera dispositivos que emulam máquinas multifuncionais e podem efetuar diferentes operações nas peças. Além disso, é considerada uma flexibilidade de rota para diferentes peças de acordo com a operação que cada peça precisa executar. Para os serviços oferecidos pela PWS, são especificadas diferentes estratégias de controle que são executadas de acordo com a escolha dos tipos de produtos. De acordo com o plano de produção e a disponibilidade das máquinas, o sistema de controle executa a alocação de recursos para a realização das operações. Considera-se que a PWS é formada por quatro máquinas (“CNC A”, “CNC B”, “RMT acb”, “RMT rcb”) as quais simulam CNCs e *Reconfigurable Machine Tools* (RMT) (KOREN; SHPITALNI, 2010). “CNC A” é capaz de realizar operações em bases “acb” (prata) e “bcb” (preta); “CNC B” realiza operações bases “rcb” (vermelha) e “bcb” (preta) e cada RMT recebe o mesmo nome das bases cilíndricas que realiza operações, isto é, RMT “acb” e RMT “rcb”. As peças deslocam-se na MPS conforme a operação do seu plano de produção considerando a ocupação da máquina. Por exemplo, “acb” é processada na “CNC A” e em seguida na “RMT acb”; a “rcb” é processada na “RMT rcb” e em seguida na “CNC B”; e “bcb” é usinada na “CNC A” quando a “CNC B” está ocupada. Após a operação de usinagem/furação a base cilíndrica é enviada para outra estação.
- (vi) Estação de controle de qualidade – *Quality Control WorkStation* (QCWS): o controle de qualidade é realizado nesta estação e é feito em todos os produtos finais. Esta estação é responsável por executar ações para garantir uma qualidade adequada. Em caso de rejeito do produto, o mesmo é marcado com a inspeção e os resultados são registrados para análise posterior.

Para cada subsistema do MPS são identificados os tipos de dispositivos, suas funções de controle, comandos e sinais de atuação e detecção. A identificação desses dispositivos é realizada de acordo com a especificação DIN/ISO 1219:1996-11 (*Deutsches Institut für Normung*)/(International Organization for Standardization) e usando os códigos recomendados na especificação IEC 61346-2:2000-12. Na Figura 50 está ilustrado um desenho da DWS e na Tabela 8 estão listados os códigos dos seus dispositivos. Por exemplo, na nomenclatura “1S2”: 1= número do circuito, S=código do dispositivo e 2=número do dispositivo.

Seguindo os passos descritos no método proposto, os modelos resultantes da aplicação de cada etapa são desenvolvidos.

Figura 50 - Desenho da DWS

Tabela 8 - Lista de dispositivos de controle da DWS.

I/O	Símbolo	Nomenclatura	Função do dispositivo	Tipo de dispositivo
I0.1	Mag_back	1B2	detecta o avanço do pistão "1A" do magazine (buffer)	sensor magnético de proximidade
I0.2	Mag_frnt	1B1	Detecta o recuo do pistão do magazine	sensor magnético de proximidade
I0.3	Sen_vac	2B1	Detecta se o vácuo está ligado	sensor de vácuo
I0.4	Arm_take	3S1	detecta se o braço giratório "2A" está na TeWS	fim de curso elétrico
I0.5	Arm_put	3S2	detecta se o braço giratório "2A" está na DWS	fim de curso elétrico
I0.6	Mat_sen	1B4	detecta se tem peça no buffer	sensor óptico
I0.7	Follow	IP_FL	detecta se a TeWS está conectada a DWS	sensor óptico
O0.0	Feed	1Y1	modifica o estado da válvula 1 para retornar pistão	solenóide
O0.1	Vacumon	2Y1	modifica o estado da válvula 2 para ativar geração de vácuo	solenóide
O0.2	Vacumoff	2Y2	modifica o estado da válvula 2 para desativar geração de vácuo	solenóide
O0.3	Armleft	3Y1	modifica o estado da válvula 3 para girar braço para DWS	solenóide
O0.4	Armright	3Y2	modifica o estado da válvula 3 para girar braço para TeWS	solenóide

5.1 MODELAGEM CONCEITUAL

Para este exemplo de aplicação, a especificação da ontologia está ilustrada na Figura 26. Os modelos utilizam estas atividades como classes de instâncias dos holons, especificando informações (física ou dados) dos dispositivos/componentes. Existem outras especificações, que serão descritas no detalhamento de cada etapa envolvendo as particulares de cada atividade do exemplo de aplicação.

No MPS identificam-se os seguintes PrHs relacionados com as peças ou produtos intermediários: $[bcb]$, $[rcb]$, $[acb]$, $[ap]$, $[bp]$, $[s]$, $[co]$, $[bcb+ap]$, $[bcb+bp]$, $[rcb+ap]$, $[rcb+bp]$, $[acb+ap]$, $[acb+bp]$, $[bcb+ap+s]$, $[bcb+bp+s]$, $[acb+ap+s]$, $[acb+bp+s]$, $[rcb+ap+s]$, $[rcb+bp+s]$ e com os produtos finais: $[bcb+ap+s+co]$, $[bcb+bp+s+co]$, $[rcb+ap+s+co]$, $[rcb+bp+s+co]$, $[acb+ap+s+co]$ e $[acb+bp+s+co]$. A Figura 51 ilustra o plano de produção no MPS com as entradas e saídas necessárias e o inter-relacionamento entre os elementos dos PrHs.

A relação entre os THs em PFS está na Figura 52 com o detalhamento do processo de negócio citado.

Figura 51 - Holons de produtos para o exemplo de aplicação em um MPS.

Figura 52 - Refinamento da atividade [processos de negócio] do holon de estratégia.

As holarquias são a seguir detalhadas. Na Figura 53 está representado o refinamento do TH-[ordem de cliente] que inclui holarquias para o gerenciamento de uma ordem de cliente e a estratégia de negociação para seleção de fornecedores. A atividade representada pelo TH-[seleção de fornecedor] representa a busca de fornecedores de produtos sob condições

aceitáveis. Se mais do que um fornecedor dispõe de uma oferta, então escolhe-se um baseado, por exemplo, no menor preço.

Figura 53 - Refinamento da estratégia de uma ordem de cliente e a seleção de fornecedores.

Cada WS fornece um serviço e a estratégia de controle de cada serviço é também representada pelo TH. A Figura 54 mostra a orquestração entre holons para obtenção do plano de produção do PrH-[$acb+bp+s+co$]. Ou seja, plano de produção envolve uma sequência de serviços representados por THs que representam as estratégias de controle para os OpHs entregarem seus respectivos serviços. Esta sequência muda de acordo com o tipo produto. O serviço da TrWS interage com todos os outros serviços para transportar a peça de uma WS para outra e, por simplificação, o fluxo entre este serviço e os demais não é representado.

A Figura 55 ilustra a estratégia de controle relacionada ao serviço da DWS. Este PFS especifica o fluxo para as peças [bcb], [rcb] e [acb] e que envolvem operações que são executadas na estação DWS. Na Figura 30 (Subseção 4.1.2) tem-se a modelagem do serviço da PWS.

Além das estratégias especificadas pelos modelos padrões, a identificação dos SuHs é também relacionada aos processos de orquestração que estão detalhados na próxima etapa (Figura 56). Do mesmo modo, na próxima etapa tem-se também a identificação de alguns dos OpHs que modelam recursos/equipamentos da MPS.

Na Figura 56 tem-se o PFS identificando as interações entre os holons (ou holarquias) relacionados ao TH-[processo de negócios] apresentado na Figura 52 e OpHs correspondentes que são requisitados por cada TH. O SuH realiza a interação e coordenação de mensagens entre os holons. Por exemplo, na Figura está representado que existe interação entre o OpH-[cliente] com um vendedor através das estratégias de controle representadas por TH-[ordem de cliente] e TH-[ordem de vendedor]; enquanto que a holarquia PrH-[plano de produção]

Figura 54 - Orquestração entre PrHs, OpHs e THs para execução de um produto.

Figura 55 - PFS do serviço disponibilizado pela estação de distribuição.

Figura 56 - Orquestração entre modelos conceituais dos processos interativos.

dos outros holons, conforme o PFS apresentado na Figura 56). Nesta Figura 57 tem-se exemplos de funções monitoração da troca de mensagens entre os holons representada por *asynchronous channel*. Usando a *IOPT web tool*, esta representação permite

[illegible]

Na Figura 58 tem-se o modelo IOPT do TH-[serviço da DWS] (Figura 55), onde os elementos IOPT representam os dispositivos identificados de acordo com os códigos listados na Tabela 8. Por exemplo, a atividade representada pelo TH-[Recua pistão] na Figura 55 é detalhada na Figura 58, onde o evento inicial corresponde à transição “1ARetreat” e o evento final corresponde à transição “ArmToDWS”. A transição “1ARetreat” tem o guard “G:_3S2=1 AND _3S1=0 AND _2B1=0”, que representa a lógica combinatória dos estados e sinais envolvidos, os quais estão de acordo com os códigos dos dispositivos listados na Tabela 8.

Figura 58 - Modelo da estratégia de controle da DWS.

A Figura 59 apresenta o modelo de um OpH do objeto de controle relativo à válvula “1Y1” que avança e recua o pistão “1A” da DWS. Os modelos tem o estado normal e/ou de falha. A disponibilidade da rede de comunicação é representada considerando se a transmissão de sinal está funcionando, ou seja, se a válvula solenoide recebeu ou não o sinal de comando. Por

TH e o OpH também interagem com outros holons, tais como quando o OpH-[cliente] é requisitado a efetuar o pagamento de acordo solicitação com o TH-[contabilidade].

Para o caso de tratamento de falhas, são apresentados os modelos dos mecanismos “diagnosticador” seguindo os passos descritos no método. De acordo com as especificações do diagnosticador, SuHs supervisionam a execução das estratégias de controle. Após modelar o funcionamento dos objetos de controle e da estratégia de controle, faz-se a descrição funcional do diagnosticador que envolve relacionar, os estados que podem ser obtidos após a execução de comandos da estratégia de controle. Em seguida os estados possíveis são correlacionados com os estados dos sensores. Sinais de comando da estratégia de controle input e output signals dos objetos de controle são conectados nesta IOPT do “diagnosticador”. Um exemplo desta modelagem está ilustrado nas Figuras 61 e Figura 62 que apresentam o diagnosticador para os atuadores da DWS. Nestes modelos, o lugar “1A Piston Retreated” é sincronizado à transição “1APistonRetreated” do “diagnosticador” por meio da “G: PistonRetreated”. Do mesmo modo, a transição “InitDiagFor1A” tem a “G:Diagn1ARetreat” para verificar os possíveis estados de falha no caso da tarefa prevista pela estratégia de controle não ter sido completada. Neste caso, os tipos de falhas considerados são: pistão “1A” está emperrado (lugar “1APistonAdvFault”) ou válvula “1Y1” (lugar “1Y1OffPistonAdvFault”) não está funcionando. Quando o diagnosticador não tem uma definição do tipo de falha, o “decisor” baseado em um procedimento de tomada de decisão define a ação a ser executada. Cada atuador da DWS tem seu sub-diagnosticador associado que é conectado ao diagnosticador, que é acionado caso a lógica “G”, que envolve sinais dos sensores, não seja atendida e indique que ocorreu alguma falha. Neste exemplo, uma ação para resolver a falha em “1A” é executada porque, em geral, um pistão tem um menor MTBF do que uma válvula solenoide ou porque se tem uma falha de comunicação da rede com esta válvula solenoide. Se a ação não resolver o problema, então uma outra ação relacionada a “1Y1” ou relacionada ao conserto dos equipamentos deve ser executada.

Os modelos em IOPT são aqui validados utilizando a ferramenta IOPT *web tool*.

A dinâmica/estrutura de sistemas multiagentes é validada com o uso de NetLogo que neste trabalho permite definir uma especificação de sistema de controle baseado em agentes, ajustando a definição de comportamento autônomo dos agentes e a interação entre eles. O código completo desta implementação está no Apêndice A.

Neste exemplo de aplicação, alguns casos são também analisados para verificar o desempenho interativo das partes do sistema de controle através da plataforma JADE+WADE.

A tela de trabalho do Netlogo com o diagrama que representa o MPS está na parte central da Figura 49. A Figura 63 mostra a tela de trabalho com as interfaces “Setup”, “Run”, “new TH-CO”, “failure-TeWS1”, “use-TeWS2-Manual”, “TeWS1-processing-time” e “TeWS2-processing-

Figura 60 - Exemplo da interação em IOPT entre TH e OpH através de asynchronous channel.

Figura 61 - Modelo diagnosticador para DWS.

Figura 62 - Modelo de sub-diagnosticador para DWS.

time” e monitores, os quais apresentam valores ou variáveis monitoradas pelo sistema, tais como o “Time...WS” (que indica o tempo de processamento de WSs), o “DefectProduct” (que indica o número de produtos manufaturados que apresentam defeitos), “PrH...” (que indica a

quantidade total de produtos finais manufaturados), e “Rest-PrH...” (que indica a quantidade restante que o sistema precisa entregar para cada tipo de produto, de acordo com as ordens de produção agendadas).

Nesta descrição no Netlogo, o MPS é considerado composto por entidades fixas, que representam as esteiras (em marrom) e as WSs (em amarelo) e entidades móveis (representam as peças e os *pallets*). Cada WS deve ser capaz de fornecer seu serviço, possui um “ID” que o identifica, possui um tempo de processamento (“processing-time”) definido em *ticks* (unidade de tempo computacional) para cada serviço e possui variáveis para simular situações do mundo real (velocidade, presença de algum defeito, direções possíveis e se existe alguma bifurcação no *layout*). Alguns serviços podem ser fornecidos por um ou mais recursos para permitir flexibilidade funcional, tais como a TeWS1 e TeWS2. Todas as WSs têm *buffers* de entrada com capacidade finita e, por simplificação, neste modelo seguem a regra *First-In First-Out* (FIFO).

Para demonstrar como é definido um *patch* (agente estacionário) neste modelo, o código na Listagem 1 demonstra como é criado o agente que representa a DWS .

Listagem 1 - Criação do agente *patch* que representa a DWS no Netlogo.

```
...
ask patch -5 0 ;; Define a coordenada em que o agente é criado.
[ set pcolor yellow ;; Define a cor do patch.
  set directions [-1 90 -1 -1] ;; Define uma lista com possíveis direções
  em relação ao eixo horizontal e que é modificada de acordo com o
  movimento na TrWS.
  set velocity 1 ;; Define a velocidade, relacionando o número de patch que
  cada agente PrH irá se mover em cada tick.
  set processing-time 20 ;; Define um tempo de processamento considerado
  para cada WS executar suas operações no PrH.
  set is-processing? FALSE ;; Se valor é "TRUE" significa que a WS está
  executando operações em um PrH, caso contrário o valor é "FALSE".
  set failure FALSE ;; Se valor é "TRUE" significa que a WS está em estado
  de falha, caso contrário o valor é "FALSE".
  set ID "DWS" ;; Define uma identificação para o agente.
  set bifurcation FALSE ;; Define se o patch está sobre uma bifurcação.
  set timeDWS 0 ;; Define o valor do instante de tempo como zero para
  contabilizar o tempo do agente.
]
...
```

Cada produto possui seu respectivo plano de produção que deve ser seguido conforme a

Figura 63 - Cenário de produção representado no Netlogo.

sequência de serviços ou escolha de parâmetros. O código da Listagem 2 demonstra como é feita a definição do plano de produção no Netlogo. Neste modelo, o usuário do sistema solicita ordens de produção através da interface “new TH-CO” incluindo requisições de produtos representados por PrHs. Após este registro, o TH-workOrder solicita aos demais holons a execução de operações e o *pallet* que transporta o PrH é identificado de acordo com o plano de produção do produto requisitado, tendo em consideração suas variáveis (por exemplo, tipo de montagem) e parâmetros de operação (por exemplo, cor da base). Além disso, devem ser definidos a capacidade do *pallet* (neste caso 1, que é o mesmo tamanho do *patch*), a direção, o tempo em *ticks* que o *pallet* entra no sistema (“*birth-time*”), e a variável “QC check” que está relacionada a existência de defeito do produto. O SuH é responsável por coordenar a agenda de execução para obtenção dos PrHs e avisa ao usuário o produto escolhido, bem como o término da execução deste produto. Na Figura 63 (na parte de baixo e a esquerda) também pode ser observado a representação do SuH e os PrHs que estão agendados para produção neste cenário.

Para garantir a execução da sequência de serviços, durante seu ciclo de vida, o PrH interage com OpHs para obter o produto de acordo com o plano de produção. Por exemplo, os agentes devem interagir para permitir mudanças de rota no OpH-TrWS em tempo de execução do

Listagem 2 - Exemplo para definição do plano de produção.

```

...
to TH-workOrder [ PrH-pallet-type ] ;;Representa a chegada de uma ordem de
  produção.
set-default-shape PrH-pallet "square" ;;Define o formato do agente PrH-
  pallet.
ask patch -5 2 ;;Requisita ao agente OpH-TrWS a condição entre [].
[ ifelse count PrH-pallet-here = 0 ;;se não houver nenhum pallet neste
  agente, execute entre [].
[ set In-Free-BufferDWS TRUE ] ;;muda o valor da variável se o buffer da
  DWS não tem um PrH-cylinder body.
[ set In-Free-BufferDWS FALSE ] ] ;;muda o valor da variável se o buffer da
  DWS tem um PrH-cylinder body.
if In-Free-BufferDWS ;;se o buffer da DWS não tem um PrH-cylinder body,
  execute entre [].
[ create-PrH-pallet 1 ;;cria um PrH-pallet, conforme condição entre []
  [ setxy -5 2 ;;no patch -5 2.
    set size 1 ;;tamanho de 1 unidade
    set heading 90 ;;direção da visão do agente, i.e., agente voltado
      para direita
    set color Blue ;;cor do agente
    set birth-time ticks; ;;considera o início do ciclo de vida do agente
      como sendo o tick em que foi criado.
    set productionPlan [ "DWS" "TeWS" "IWS" "SWS" "PWS" "AWS" "QCWS" "OUT
      " ] ;;define o plano de produção inicial do PrH que está entre []
      set processing FALSE ;;inicia o valor da variável 'processing',
        para monitorar se WS está processando o PrH
    set QC-check "GOOD" ;;inicia o valor da variável 'QC-check', para
      monitorar defeitos no PrH.
    set priority 0 ;;inicia o valor da variável 'priority', para
      monitorar a prioridade de execução do PrH
  ]
]
...

```

PrH. Este controle de rota é realizado monitorando certas condições para mudar a velocidade, a distância e a direção do PrH-*pallet*, conforme demonstra o exemplo na Listagem 3.

Quando é necessária uma reconfiguração por ocorrência de uma falha, o SuH modifica o plano de produção e supervisiona o comportamento da negociação entre os holons. O código da Listagem 4 demonstra parte dessa negociação. Neste exemplo é mostrado quando o próximo serviço a ser executado é aquele fornecido por uma TeWS. Existem duas WSs que podem

Listagem 3 - Supervisão em tempo de execução durante movimentação de PrHs na TrWS.

```

...
to TH-TrWS-pallet ;;Controla o movimento do pallet
ask PrH-pallet ;;requisita ao PrH-pallet entre []
[ if priority = 0 ;;se a variável ''priority'' = 0, execute entre
  []
  [ ask patch-here ;;requisita ao patch atual o que está entre []
    [ set distance-to-jump velocity ;;muda a velocidade do PrH
      set new_directions remove -1 directions ;;define qual a
        direção do PrH, no atual patch.
    ]
  ]
...

```

realizar o serviço relacionado (TeWS1 e TeWS2), em uma bifurcação (chamada de “InTeWS”) e o sistema deve decidir para qual WS deve ser direcionado o PrH. Neste modelo, o PrH será direcionado para TeWS1 se não houver falha na mesma e se a TeWS2 não estiver liberada para uso manual; e caso uma dessas condições seja verdadeira, o PrH é enviado para a TeWS2. Este modelo de SRM também modifica o tempo de processamento de WSs, para reconfigurar a produção conforme mudanças de demanda (uma parte do código para esta modificação está na Listagem 5).

Para descrever como é feito o mapeamento entre o modelo IOPT e o Netlogo, a Figura 64 mostra um modelo IOPT (no ambiente da IOPT *webtool*) do TH que representa a estratégia de controle usando o código acima citado na denominação dos `lugares` e `transições`. Ou seja, os PrHs são processados na TeWS2 caso ocorra falha na TeWS1, ou se a TeWS2 não está processando nenhum *pallet* e ela está liberada para uso (condição dada pela variável “use-TeWS2-Manual”). Ao cumprir o processamento da referida estação o plano de produção é modificado, incluindo (“fput”) ou removendo (“remove”) os serviços a serem executados. Essa figura também ilustra os modelos IOPT do SuH responsável pelo transporte, OpH que representa a TeWS e OpH que representa uma parte da TrWS.

No Netlogo, o OpH espera ser solicitado a realizar uma operação, e isto é indicado com a chegada de um *pallet* na entrada de uma WS. Quando o *pallet* se posiciona no local especificado e relativo ao OpH, este agente muda seu estado para indisponível (através da variável “is-processing”) e executa a operação programada. Quando o processamento estiver concluído, o OpH notifica o resultado da execução (sucesso ou falha) ao PrH e fica novamente disponível para executar uma nova operação.

A IWS e a QCWS são responsáveis pela verificação de qualidade dos produtos sendo que

Listagem 4 - Negociação entre holons para mudança do plano de produção e atendimento de novas demandas.

```

...
    if not processing ; condição para o agente requisitar o OpH-TeWS
    and first productionPlan = "TeWS" ; condição de verificação do plano
      de produção
    and [ID] of patch-here = "InTeWS" ; condição se o patch atual é a
      bifurcação que representa a entrada da TeWS, execute []
    [ if not [failure-TeWS1] of patch-at 2 2 ; condição se não houver
      falha na TeWS1
    and not use-TeWS2-Manual ; condição se a TeWS2 estiver liberada para
      uso em seleção manual
    [ set heading item 0 directions ; põe o agente PrH-pallet na 1a
      direção da lista , neste caso , vertical de baixo para cima
    if not any? PrH-pallet-on patch-ahead distance-to-jump ; verifica se
      tem algum PrH-pallet no próximo patch
    [ jump distance-to-jump ; desloca o agente da distância pré-definida
    set productionPlan remove "TeWS" productionPlan ; modifica o plano de
      produção retirando do plano o serviço de eTeWS
    set productionPlan fput "TeWS1" productionPlan ; inclui no plano o
      serviço de TeWS1
    ]
  ]

    if [failure-TeWS1] of patch-at 2 2 ; se houver uma falha na TeWS
      execute entre []
    [ set heading item 2 directions ; mude direção do OpH-Pallet para
      vertical sentido de cima para paixo
    if not any? PrH-pallet-on patch-ahead distance-to-jump ; verifica se
      tem algum OpH-Pallet no próximo patch
    [ jump distance-to-jump ; desloca o OpH-pallet da distância pré-
      definida
    set productionPlan remove "TeWS" productionPlan ; modifica o plano
      de produção retirando do plano o serviço de TeWs
    set productionPlan fput "TeWS2" productionPlan ; inclui no plano o
      serviço de TeWS2
    ]
  ]
...

```

neste caso, para fins de análise do sistema, adota-se que a ocorrência de problemas de qualidade ocorrem de uma forma aleatória, notificando os resultados para o SuH. Se o número aleatório

Listagem 5 - Negociação entre holons para atendimento de novas demandas.

```

...
to SuH-WsProcessing
ask patches
[ if ID = "TeWS1" [ set processing-time TeWS1-processing-time ]
if ID = "TeWS2" [ set processing-time TeWS2-processing-time ]
if ID = "CNC A" [ ifelse ( item 0 quaAcb + item 1 quaAcb ) < 50
[ set processing-time 40 ] [ set processing-time 20 ] ]
if ID = "CNC B"
[ ifelse ( item 0 quaRcb + item 1 quaRcb ) < 50
[ set processing-time 40 ] [ set processing-time 20 ] ] ] ]
...

```

(escolhido entre 0 e 1) for menor que "0.1", o produto é considerado com defeito. O código para esta verificação na QCWS, que é similar ao da IWS está na Listagem 6

Listagem 6 - Código para implementação do controle de qualidade no Netlogo.

```

ask pallet
...
if [ID] of patch-here = "QCWS" ;;verificação de controle de qualidade
[ ifelse random-float 1 < 0.1 ;;definição de função aleatória
[ set QC-check "BAD" ;;se a condição após ifelse for verdadeira altera a
variável QC-check para BAD
]
[ set QC-check "GOOD" ;;se a condição após ifelse for falsa altera a
variável QC-check para GOOD
]
]
...

```

Neste exemplo de aplicação, se inclui particularidades como: capacidade de fabricar vários tipos de produtos com diferentes matérias-primas, flexibilidade/mudança de rota, WSs com comportamento específico para considerar reconfiguração, tal como a PWS, gráfico e variáveis para monitorar o tempo de ocupação por WS. Outras especificações são consideradas no uso do Netlogo:

- (i) a análise do sistema é baseada em um tempo de processamento para cada WS: de 50 *ticks* para AWS, ajustável através de *sliders* (botão do Netlogo que permite mudar o valor de uma variável dentro de certo intervalo) em 20 ou 40 *ticks* para TeWS1 e TeWS2, e

Figura 64 - Exemplo de modelo em IOPT usado para descrição do mapeamento com o Netlogo.

de 20 *ticks* para as demais WSs. Considera-se também que cada sub-WS da PWS tem um tempo de 20 *ticks* cada uma para realizar suas atividades;

- (ii) para permitir análise de outros cenários, interfaces para simular mudanças no sistema são consideradas:
 - (a) simulação de uma quebra da TeWS1. Neste caso, automaticamente o sistema deve permitir o uso da TeWS2;
 - (b) permissão ou não de operação manual da TeWS2; e
 - (c) escolha de tipos de produtos finais. Por exemplo, é possível escolher para cada ordem

de cliente a cor da base cilíndrica: vermelha, preta, prata ou todas. Caso a DWS alimente o sistema com uma base cilíndrica diferente da escolhida do conjunto das ordens de clientes, o SWS deve descartá-la.

- (iii) permissão de criação de PrHs com planos pré-definidos de produção que contém estratégias pré-definidas de controle. Durante a execução, o SuH interage com OpH e PrH de forma a garantir a execução ou reconfiguração do plano de produção pré-definido para atender novas demandas. Um fragmento do código que permite a reconfiguração baseado na mudança de produto é apresentado na Listagem 7; e

Listagem 7 - Código para implementação da mudança de produto no Netlogo.

```
ask pallet [
...
if ( item 0 quaRcb > 0 or item 1 quaRcb > 0 ); verifica se
    quantidade de PrH-[rcb+bp] é diferente de zero
and color = red ; verifica cor da base cilíndrica no pallet
[ jump distance-to-jump
set productionPlan remove-item 0 productionPlan
set processing TRUE    set is-processing? TRUE
set time-of-processing processing-time
set TimeSWS TimeSWS + processing-time
]; se condição for verdadeira a peça continua no sistema
if ( item 0 quaRcb <= 0 and item 1 quaRcb <= 0 );; verifica
    se quantidade de PrH-[rcb+bp] é igual a zero
and color = red ; se condição for verdadeira, muda o plano
    de produção para descartar a base cilíndrica
[ jump distance-to-jump
set productionPlan fput "SWSOUT" productionPlan
set productionPlan remove "PWS" productionPlan
set productionPlan remove "AWS" productionPlan
set productionPlan remove "QCWS" productionPlan
set productionPlan remove "OUT" productionPlan
set processing TRUE    set is-processing? TRUE
set time-of-processing processing-time
set TimeSWS TimeSWS + processing-time
set heading item 2 directions ]
...
]
```

- (iv) visualização de resultados por meio de processos de monitoração:

- (a) da evolução de duas variáveis para análise de desempenho de sistemas de manufatura:

- o *Work In Process* (WIP) e a média do *Manufacturing Lead Time* (MLT). Além disso, é visualizado a evolução do número de produtos finalizados com sucesso ou com defeito. Estas variáveis são monitoradas de acordo com o código da Listagem 8;
- (b) da evolução da variável “use-TeWS2-Automatic” que serve para observar a execução automática de reconfiguração no caso de quebra da Te-WS1;
 - (c) da evolução do tempo de ocupação por WS para identificar os gargalos de produção. Este tempo corresponde a soma de cada ciclo executado na operação da WS por peça manufaturada. Por exemplo, a variável “TimeDWS” na Figura 63 indica 460 *ticks* que representa a soma do tempo de processamento que a DWS executou operações nas bases cilíndricas na simulação deste cenário que foi de 1000 *ticks*. Parte do código para verificar o tempo de execução é dado na Listagem 9.

Diferentes cenários de produção são assim simulados usando o Netlogo. A Figura 63 ilustra um cenário com os seguintes parâmetros: escolha de produtos [bcb/rcb/acb+ap+s+co] (isto é, esta configuração permite obter três diferentes produtos usando todas as bases cilíndricas), permite o uso de TeWS2, tempo de processamento de TeWS1 e TeWS-2 de 40 *ticks* cada uma, e nenhuma ocorrência de falha em TeWS1. Um resultado gráfico obtido com esta configuração é ilustrado na parte de cima e a esquerda na Figura 66 que mostra outros três cenários. Os gráficos demonstram a soma do tempo de processamento de cada WS, de acordo com a legenda. Para visualizar os resultados, intervalos são diferentes para cada WS no eixo “x”, conforme indicado na Figura 65; entretanto, todas as simulações correspondem ao número atual de *ticks* da simulação no eixo x. Para permitir uma análise comparativa, os cenários das simulações apresentadas correspondem a 1000 *ticks*.

Na parte de cima dos gráficos de cada cenário estão as interfaces de cada parâmetro de configuração do sistema e abaixo dos gráficos estão os monitores das variáveis escolhidas. Na parte de baixo de cada gráfico, estão elementos para monitorar variáveis de evolução da produção. O Cenário 1 é configurado para obter três produtos e neste caso a TeWS1 e TeWS2 são programadas com o tempo de processamento de 20 *ticks* por base cilíndrica. Os resultados obtidos neste cenário indicam que a alocação de recursos está bem balanceada, incluindo as sub-WSs de PWS na qual todas as máquinas são usadas em um tempo de ocupação similar. O cenário 2 (na parte de cima e a direita na Figura 66) apresenta resultados similares em MLT e em quantidade de produtos finais; entretanto, a TeWS1 e TeWS2 têm um tempo de ocupação maior porque foram configuradas com 40 *ticks*. Isto demonstra que é possível reduzir o tempo de processamento desta WS sem grandes perdas de produção, obviamente dependendo do tempo de produção. O Cenário 3 (na parte de baixo e a esquerda na Figura 66) é de manufatura de um tipo de produto [acb+bp+s+co] e a quantidade de produtos finais é

Listagem 8 - Código para monitoração de *Work In Process* e *Manufacturing Lead Time*.

```

....
;;;monitora a quantidade de produto prh-bcb-ap-sp-co produzida no sistema
to SuH-countOutput
ask PrH-pallet
[ if [ID] of patch-here = "OUT"
[ set death-time ticks
  ifelse QC-check = "GOOD"
  [ set FinalProduct-outputTotal FinalProduct-outputTotal + 1
    if shape = "prh-cb-ap-sp-co" and color = cyan
    [ set FinalProduct-outputPrH_bcb_ap FinalProduct-outputPrH_bcb_ap +
      1
      let quaBcbAp item 0 quaBcb - 1
      set quaBcb replace-item 0 quaBcb quaBcbAp
    ]
  ]
...
;;;;monitora MLT
[ set FinalProduct-output-bad FinalProduct-output-bad + 1
]
set mlt-aux (mlt-aux + (death-time - birth-time))
set avg-mlt (mlt-aux / (FinalProduct-outputTotal + FinalProduct-output-bad))
...
;;;;;monitora WiP
to SuH-statistics
set wip count PrH-pallet-on patches + count PrH-cylinderBody-on patches
...
end
...

```

menor do que nos cenários 1 e 2. A causa é atribuída ao fato da DWS ter de verificar a cor da base cilíndrica e parte do tempo gasto pelo sistema é dedicado a descartar bases diferentes de PrH-[acb]. No cenário 4 (na parte de baixo e a direita na Figura 66), considera-se a ocorrência de um defeito em TeWS1 e automaticamente o sistema usa TeWS2 (vide o valor "true" no monitor correspondente). Estes cenários ilustram alguns casos que o Netlogo pode ser usado para analisar aspectos importantes do controle de SRMs.

Listagem 9 - Verificação do tempo de execução em WorkStation.

```

;;;;; monitora tempo de processamento
to statistics
...
set wip count PrH-pallet-on patches + count PrH-cylinderBody-on patches
set-current-plot-pen "DWS"
plot TimeDWS
set-current-plot-pen "TeWS1"
plot TimeTeWS1
set-current-plot-pen "TeWS2"
plot TimeTeWS2
set-current-plot-pen "IWS"
plot TimeIWS
set-current-plot-pen "SWS"
plot TimeSWS
set-current-plot-pen "RMT rcb"
plot TimeRMT_rcb
set-current-plot-pen "RMT acb"
plot TimeRMT_acb
set-current-plot-pen "CNC A"
plot TimeCNC_A
set-current-plot-pen "CNC B"
plot TimeCNC_B
set-current-plot-pen "AWS"
plot TimeAWS
set-current-plot-pen "QCWS"
plot TimeQCWS
set-current-plot-pen "TePaIWS"
plot TimeTePaIWS
end
...

```

5.3 IMPLEMENTAÇÃO

Os códigos Java derivados dos modelos funcionais são gerados usando JADE+WADE. Na Figura 67 é apresentada a tela desta plataforma usada para a edição e implementação da estratégia de controle [seleção de fornecedor] (ilustrada na Figura 53) e indica a codificação gerada a partir do *workflow*. Abaixo e a esquerda da Figura 67, tem-se um exemplo do GUI Sniffer do JADE, que serve para monitorar a troca de mensagens entre os agentes.

Baseado no modelo da Figura 58, a IOPT *web tool* gera os arquivos do programa de

Figura 65 - Tela do Netlogo usada para configurar o gráfico do tempo de ocupação.

controle da DWS e um fragmento do arquivo “exec step.c” para a transição StartDWS é apresentado na Listagem 10.

A Tabela 8 também lista os endereços dos sensores e detectores relacionando-os com as portas I/O do controlador industrial e a Figura 68 ilustra um SFC para controlar a DWS considerando estes endereços.

5.4 OBSERVAÇÕES ADICIONAIS

O exemplo de aplicação ilustra os principais aspectos do método proposto para a implementação do controle de um sistema reconfigurável de manufatura. Alguns indicadores quantitativos são usados para comparação de diferentes arquiteturas de controle, como por exemplo, o *lead time* (isto é, o tempo total requerido para entrega de um produto), o número de peças produzidas, o tempo necessário para executar um ciclo de produção e a utilização de recursos (isto é, o tempo de ocupação dos recursos do sistema). Neste exemplo de

Figura 66 - Simulação de cenários de produção no Netlogo.

aplicação, a reconfiguração é considerada em diversas formas seja através da adição, remoção ou substituição de recursos; seja na consideração de novas demandas de produção ou seja para aumentar a segurança em caso de quebra de equipamentos. Apresenta-se como diferentes cenários podem ser analisados e como o controle de SRM respondeu com comportamento colaborativo, confirmando que o controle efetivamente atua para reconfigurar o sistema tanto

Figura 67 - Tela do JADE+WADE da atividade [seleção de fornecedor].

Listagem 10 - Fragmento de arquivo do programa de controle da DWS.

```

/* Transition 394 - StartDWS */
int t_394_enabled( empty_NetMarking* prev,
empty_NetMarking* avail )
{
    return ( avail->p_185 >= 1 );
}
int t_394_events( empty_InputSignalEvents* events )
{
    return ( events->StartOn );
}
int t_394_guards( empty_NetMarking* marking,
empty_InputSignals* inputs,
empty_PlaceOutputSignals* place_out,
empty_EventOutputSignals* ev_out )
{
    return ( ev_out->_1B1 == 0 && ev_out->_1B2 == 1 && ev_out->_1B4
== 0 && ev_out->_2B1 == 0 && ev_out->_3S1 == 1 && ev_out->_3S2 ==
0 && inputs->_4B2 == 1 );
}
void t_394_remove_marks( empty_NetMarking* marking )
{
    marking->p_185--;
}
void t_394_add_marks( empty_NetMarking* marking )
{
    marking->p_11++;
}
void t_394_generate_output_events( empty_OutputSignalEvents* ev_out )

```

em situação de falhas como em novas demandas. Vantagens operacionais incluem melhor e mais eficiente uso de recursos de manufatura, velocidade de produção e capacidade de entregar

Figura 68 - SFC para controle da DWS

produtos personalizados.

Várias razões contribuíram para a escolha deste exemplo de aplicação. Trata-se de um equipamento dedicado a treinamento prático, porém o MPS contém os mesmos dispositivos utilizados em aplicações industriais. Por ser concebido para fins educacionais, o fabricante disponibiliza uma vasta documentação na internet (MPS, 2015). O MPS atende todas as características para SRMs. O sistema trabalha com uma única família de produto mas possibilita a personalização dos produtos. As estações de trabalho PWS e a AWS podem modificar suas funcionalidades para se adequarem a novos requisitos de produtos (convertibilidade). O sistema é dividido em estações de trabalho (WSs) as quais podem ser organizadas para formar diferentes esquemas de produção (modularidade). Estes esquemas são integrados por um conjunto de dispositivos mecatrônicos, interfaces para controle de processos ou de dispositivos que asseguram a integrabilidade do sistema como um todo. As estações de trabalho IWS e QCWS têm dispositivos para identificar defeitos em produtos. Estas

funcionalidades são consideradas na concepção do sistema de controle que envolve mecanismos de tolerância a falhas assegurando a diagnosticabilidade do sistema. Dispositivos de controle de mecanismos são disponibilizados para este sistema operar com autonomia. Todas as WSs têm um controlador industrial Siemens Simatic S7-300, os quais são conectados a internet por meio da rede Ethernet, permitindo operadores e clientes interagirem com o MPS, isto é, transferindo informações para o controlador via um terminal *Profibus DP*, para o qual todas as entradas e saídas de processos são conectadas (interoperabilidade). A arquitetura de controle proposta considera plataformas de HMAS as quais permitem operar em sistemas heterogêneos através de uma *Java Virtual Machine*, o que garante a portabilidade. Além disso, a abordagem desenvolvida na TrWS considerando segmentos de transportes representados por componentes colaborativos e autônomos demonstra potencial para estender a proposta e desenvolver sistemas de manufatura com *layouts* reconfiguráveis.

A especificação dos componentes da arquitetura de controle proposta consideram os diferentes cenários produtivos onde aspectos de reconfiguração devem ser tratados. Essas especificações diretamente derivadas dos modelos são concebidos usando o paradigma de sistemas holônicos e multiagentes, em uma abordagem modular e flexível, que segue conceitos de arquitetura orientada a serviços. O exemplo de aplicação apresenta a modelagem e a especificação desses componentes sob o enfoque de projeto do produto, recurso ou funcionalidade.

A comparação das condições e ações que precisam ser atendidas em cada serviço fornecido por um holon/agente é descrito usando elementos da IOPT que também permitem a composição dos modelos de forma clara e condensada. Desse modo, tem-se a reconfiguração no caso de necessidade de mudança de alguma programação para atender uma nova demanda ou situação de falha. Os modelos são considerados também para sistematizar a identificação de falha, onde os processos de diagnóstico e decisão são devidamente especificados. Portanto, a reconfiguração neste exemplo de aplicação é aplicada para aumento ou redução da produção que também é um tipo de flexibilidade. Os cenários tratados nas ferramentas demonstram alguns resultados práticos.

O Netlogo possui recursos para que se possa analisar diferentes configurações de manufatura. O sistema analisado com o Netlogo aceita ordens personalizadas em tempo real e modifica os planos de produção de acordo com a demanda ou ocorrência de falha. A implementação de atividades com o uso da plataforma JADE+WADE comprova como os programas de controle podem ser facilmente obtidos. Os modelos em IOPT mostraram ser efetivos para análise funcional e também facilitam a geração dos programas dos controladores programáveis industriais. Ou seja, os modelos desenvolvidos demonstram que agentes podem negociar entre eles para atender o plano final.

Testes de aplicação do método proposto confirmam seu potencial para a concepção do controle de sistemas reconfiguráveis de manufatura. Através da análise via simulação dos processos obtêm-se, conclusões do comportamento e que podem ser usadas para o refinamento dos parâmetros na especificação do sistema de controle.

6 CONCLUSÕES

Este trabalho propõe uma arquitetura de controle e um método para desenvolvimento de seus componentes para implementar um Sistema Reconfigurável de Manufatura. A arquitetura de controle proposta é baseada em HMAS e SOA, e as funcionalidades dessa arquitetura são desenvolvidas usando as técnicas do PFS e IOPT. A ferramenta IOPT *web tool*, a plataforma de HMAS JADE+WADE e o *software* de simulação de sistema multi-agente Netlogo são considerados para análise, verificação e implementação do controle de SRM. IOPT *web tool* e JADE+WADE são adotadas por suas vantagens no desenvolvimento de sistemas, tais como: facilitar a validação/obtenção do programa de controladores; considerar características de interoperabilidade, portabilidade e robustez; estarem disponíveis via *web* como *software* abertos; considerar extensões que permitem desenvolver códigos a partir de *workflow*; e considerar a decomposição de sistemas GALS, permitindo obter a validação de cada processo de forma global e individual.

Apesar de ainda ser pouco utilizado para desenvolvimento de sistemas de controle, o Netlogo apresenta grande potencial para desenvolver sistemas de controle baseados em HMAS e é adotado visto que apresentou melhores resultados em estudos comparativos, destacando-se sua disponibilidade de suporte/documentação, facilidade de programação e interface amigável.

A solução proposta:

- (i) propõe mecanismos para a composição de uma estrutura de controle baseada em holarquias, considerando características de autonomia, recursão e integração através da interação entre os holons. É descrito como um holon pode pertencer a diferentes holarquias tanto do mesmo nível de controle como de níveis diferentes;
- (ii) facilita e sistematiza a implementação da solução de controle, adequando uma sistemática para projeto e comunicação entre equipes. No método de desenvolvimento, o PFS é usado para a modelagem conceitual dos componentes de controle. A IOPT é adotada para análise e especificação das funcionalidades do sistema. JADE+WADE permite desenvolver a codificação de programas do sistema de controle usando *workflow*. Ou seja, a abordagem assegura a integração dos diferentes níveis do sistema de controle; desde processos de negócio até estratégias de controle no nível de chão-de-fábrica;
- (iii) combina HMAS e SOA por meio do conceito de encapsulamento relacionado às especificações derivadas dos modelos. Além disso, JADE+WADE disponibiliza rotinas para os agentes requisitarem e acessarem serviços distribuídos e dispersos por meio de *Web services*; e

- (iv) permite a reconfiguração do sistema de manufatura de acordo com estratégias de controle e coordenação das operações, registro das habilidades de cada holon, e modificação das holarquias em resposta a mudanças de demanda ou ocorrência de falhas.

Os modelos conceituais e funcionais dos componentes da arquitetura de controle são desenvolvidos com base em três instâncias do controle de um sistema reconfigurável de manufatura: processos, produtos e dispositivos. Além disso, os modelos consideram um tipo de comportamento social, isto é, os comportamentos dos componentes devem ser colaborativos e o sistema de controle deve considerar a participação humana no processo industrial. Esta forma de flexibilidade de projeto aliada com a proposta de especificação de uma ontologia garante um adequado modelo semântico à proposta, isto é, garante que a especificação das partes do sistema de controle seja desenvolvida de modo sistematizado combinando conceitos da abordagem *top down* com a abordagem *bottom-up*. Como a predição de todos os possíveis estados e comportamentos futuros é uma tarefa praticamente impossível, o método considera o uso de ferramentas de análise e simulação de cenários como a *IOPT web tool* e o *software* Netlogo para verificação, validação e análise dos resultados. O modelo proposto foi implementado em um sistema real de manufatura onde diferentes cenários permitiram o desenvolvimento da abordagem e análises relacionadas a escalabilidade e robustez. Além disso, o tempo requerido para o projeto é reduzido através do uso de ferramentas acima citadas, disponíveis e abertas à comunidade industrial/científica.

Portanto, a proposta considera características de reconfiguração de processos, dispersão, e integração homem-máquina e outras boas práticas para desenvolver um efetivo controle de um Sistema Reconfigurável de Manufatura. Ou seja, para desenvolver esta solução, os principais requisitos necessários ao controle de um SRM são considerados e confirmados pelos resultados apresentados no exemplo de aplicação:

- (i) assegurar agilidade e flexibilidade de adaptar o sistema a novas demandas da planta industrial, isto é, o sistema de controle de SRM deve adaptar-se a novos produtos, mudanças de quantidade de produtos, integração de novas tecnologias/processos e situações causadas por ocorrência de falhas;
- (ii) permitir o acesso a informações detalhadas de cada processo do produto e com isto assegurar maior interação entre clientes e fornecedores para atender a demandas individuais;
- (iii) assegurar responsividade cobrindo as características de personalização, convertibilidade, escalabilidade, modularidade, integrabilidade e diagnosticabilidade;

- (iv) adotar uma técnica de modelagem composta por módulos distribuídos que colaboram entre si com certa autonomia e inteligência combinados com uma supervisão global;
- (v) sistematizar o desenvolvimento destes módulos para permitir a comunicação/desenvolvimento entre diferentes equipes e facilitar o reuso; e
- (vi) integrar homem e máquina na modelagem funcional onde atividades operacionais são descritas como serviços.

6.1 TRABALHOS FUTUROS

A definição da ontologia é fundamental para correta comunicação da equipe/entidades envolvidas e permitir colaboração entre sistemas baseados em holons e agentes. Neste sentido, o trabalho apresenta também uma proposta para definição da ontologia integrada ao método de desenvolvimento da arquitetura de controle. Entretanto, a normatização de termos e abordagens é ainda considerada um grande desafio no projeto de sistemas de controle e propor extensões das contribuições propostas aqui às normas existentes é um dos trabalhos que facilitarão a adoção de novos paradigmas na indústria.

A validação da proposta em outros casos/estruturas de plantas industriais deve ser considerada. A proposta apresenta características inovadoras e pode ser considerada um método para a concepção e análise de sistemas de controle de outros paradigmas que necessitam considerar reconfiguração em seus processos. Por exemplo, a proposta pode ser aplicada para sistemas de controle da Indústria 4.0 considerando-se nos modelos tecnologias como o RFID em cada item do fluxo de materiais e sistemas *smart* em certos processos produtivos. Por outro lado, visto que sistemas baseados em agentes necessitam trocar grande quantidade de informações, a Indústria 4.0 mostra-se como uma efetiva abordagem para se contemplar os objetivos dos Sistemas Reconfiguráveis de Manufatura, pois considera a dispersão/integração de *cyber physical systems* e sistemas inteligentes em uma infraestrutura de computação na nuvem que deve processar quantidade relativamente grande de dados. Outra perspectiva é a integração de sistemas autônomos e inteligentes com estratégias de controle que permitam o desenvolvimento/melhoria de indicadores de desempenho para sistemas/produtos que considerem sustentabilidade em todos os seus processos.

Apesar das ferramentas propostas aqui mostrarem-se efetivas para o desenvolvimento do controle de SRMs. Um estudo comparativo deve ser objeto de revisão em futuras aplicações, pois as ferramentas estão em constante evolução. Por exemplo, observa-se que futuras versões do Netlogo permitirão a simulação de HMAS em ambiente tridimensional e que existem propostas de desenvolvimento de JADE+WADE que permitirão a integração através de sistema

operacional de dispositivo *smart*.

Considerando-se a grande variedade de falhas e suas consequências, os modelos podem ser ampliados para implementar mecanismos de controle que permitam evitar/corrigir ruptura na cadeia de suprimento e implementar barreiras de segurança para evitar acidentes. São exemplos de falhas que podem ser consideradas nos modelos: (i) rupturas na cadeia de suprimento, como a falta de um tipo de matéria-prima no início da execução; (ii) quebra ou mal-funcionamento do subsistema de transporte; (iii) falha de sensores; e (iv) falhas críticas as quais podem afetar um sistema de manufatura causando acidentes graves.

A implementação pode ser ampliada considerando-se outros parâmetros estocásticos. Apesar dos modelos e as ferramentas usadas considerarem a aleatoriedade de sistemas reais (tais como, no desenvolvimento do controle de qualidade, na inserção de novas demandas, na ocorrência de falhas e no fornecimento de matéria-prima), os modelos propostos podem ser refinados usando-se técnicas de estatística. Por exemplo, o tratamento estatístico de dados de entrada em cada modelo de componente pode ser considerado, assim como dados estatísticos podem ser incluídos no proposto mecanismo de controle da produção baseado em restrição temporal.

Referências

ADAM, N. R.; ATLURI, V.; HUANG, W.-K. Modeling and analysis of workflows using Petri nets. **Journal of Intelligent Information System**, Boston, Kluwer Academic Publisher, v. 10, n. 2, p. 131–158, mar 1998. DOI: 10.1023/A:1008656726700.

ALI, S. A.; SEIFODDINI, H.; SUN, H. Intelligent modeling and simulation of flexible assembly systems. **Proceedings of the 37th Conference on Winter Simulation**. p. 1350–1358, 2005.

ANSI/ISA. **Batch Control Part 1: Models and Terminology, American National Standard**. Research Triangle Park, NC: ANSI/ISA-88.01, 1995.

ARAKAKI, J.; MIYAGI, P. E. Degeneration methods in intelligent building control system design. **Information Technology For Balanced Manufacturing Systems**, Springer, p. 469–478, 2006.

ASATO, O. L. et al. Control of productive systems with functional flexibility level. **IEEE 16th Conference on Emerging Technologies & Factory Automation (ETFA)**. p. 1–4, 2011.

ASATO, O. L. et al. Using the Enhanced-Mark Flow Graph for dynamic resource allocation in distributed manufacturing. **International Journal of Computer Integrated Manufacturing**, Taylor & Francis, p. 1–14, 2015.

AVIZIENIS, A. et al. Fundamental concepts of dependability. **Technical Report Series**, University of Newcastle Upon Tyne, 2001.

BAL, M.; HASHEMIPOUR, M. Virtual factory approach for implementation of holonic control in industrial applications: a case study in die-casting industry. **Robotics and Computer-Integrated Manufacturing**, Elsevier, v. 25, n. 3, p. 570–581, 2009.

BARBOSA, J.; LEITÃO, P. Simulation of multi-agent manufacturing systems using agent-based modelling platforms. **9th IEEE International Conference on Industrial Informatics (INDIN)**. p. 477–482, 2011.

BARBOSA, J. et al. Self-organized holonic manufacturing systems combining adaptation and performance optimization. In: **Technological Innovation for Value Creation**. Berlin: Springer, 2012. p. 163–170.

BARBOSA, J. et al. Enhancing ADACOR with biology insights towards reconfigurable manufacturing systems. **37th Annual Conference on IEEE Industrial Electronics Society (IECON)**. p. 2746–2751, 2011.

BECHHOFFER, S. OWL: Web Ontology Language. In: **Encyclopedia of Database Systems**. Berlin: Springer, 2009. p. 2008–2009.

BÉHÉ, F. et al. An ontology-based metamodel for multiagent-based simulations. **Simulation Modelling Practice and Theory**, Elsevier, v. 40, p. 64–85, 2014.

BELLIFEMINE, F.; POGGI, A.; RIMASSA, G. JADE – A FIPA compliant agent framework. **4th International Conference on Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM)**. v. 99, p. 97–108, 1999.

BELLIFEMINE, F. L.; CAIRE, G.; GREENWOOD, D. **Developing multi-agent systems with JADE**. Hoboken, Nova Jersey: Wiley, 2007. v. 7.

BI, Z. et al. Reconfigurable manufacturing systems: the state of the art. **International Journal of Production Research**, Taylor & Francis, v. 46, n. 4, p. 967–992, 2008.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **Unified Modeling Language User Guide**. Boston: Addison-Wesley, 2005.

BORANGIU, T. et al. An implementing framework for holonic manufacturing control with multiple robot-vision stations. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 22, n. 4, p. 505–521, 2009.

BOTTI, V.; BOGGINO, A. G. **ANEMONA: A multi-agent methodology for holonic manufacturing systems**. Berlin: Springer, 2008.

BRENNAN, R.; FLETCHER, M.; NORRIE, D. An agent-based approach to reconfiguration of real-time distributed control systems. **IEEE Transactions on Robotics and Automation**, IEEE, v. 18, n. 4, p. 444–451, 2002.

BRENNAN, R. W.; NORRIE, D. H. The performance of partial dynamic hierarchies for manufacturing. **Proceedings of the Second International Workshop on Intelligent Manufacturing Systems, Katholieke Universiteit Leuven, Leuven, Belgium**. p. 5–13, 1999.

CAIRE, G.; GOTTA, D.; BANZI, M. WADE: A software platform to develop mission critical applications exploiting agents and workflows. **7th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS): Industrial Track**. p. 29–36, 2008.

CAIRE, G. et al. Wolf – An Eclipse Plug-in for WADE. **17th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. WETICE'08**. p. 26–32, 2008.

CANDIDO, G. et al. Applications of dynamic deployment of services in industrial automation. In: **Emerging Trends in Technological Innovation**. Berlin: Springer, 2010. p. 151–158.

CARDOSO, J.; VALETTE, R. **Redes de Petri**. Santa Catarina: Ed. UFSC, 1997.

CASSANDRAS, C. G.; LAFORTUNE, S. **Introduction to discrete event systems**. Berlin: Springer, 2008.

CASTLE, C. J.; CROOKS, A. T. Principles and concepts of agent-based modelling for developing geospatial simulations. **CASA Working Papers**, Centre for Advanced Spatial Analysis, University College London (UCL), v. 110, 2006.

CAVALIERI, S. et al. An experimental benchmarking of two multi-agent architectures for production scheduling and control. **Computers in Industry**, Elsevier, v. 43, n. 2, p. 139–152, 2000.

CERNUZZI, L.; COSSENTINO, M.; ZAMBONELLI, F. Process models for agent-based development. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 18, n. 2, p. 205–222, 2005.

CHEN, C.-H.; OCCEÑA, L. G. Knowledge decomposition for a product design blackboard expert system. **Artificial Intelligence in Engineering**, Elsevier, v. 14, n. 1, p. 71–82, 2000.

CHIRN, J.; MCFARLANE, D. A holonic component-based approach to reconfigurable manufacturing control architecture. **11th International Workshop on Database and Expert Systems Applications**. p. 219–223, 2000.

CHRISTENSEN, J. HMS/FB architecture and its implementation. In: **Agent-Based Manufacturing. Advances in the Holonic Approach**, Berlin, Germany: Springer-Verlag, p. 53–88, 2003.

COLOMBO, A. W.; HARRISON, R. Modular and collaborative automation: achieving manufacturing flexibility and reconfigurability. **International Journal of Manufacturing Technology and Management**, Inderscience, v. 14, n. 3, p. 249–265, 2008.

COLOMBO, A. W.; NEUBERT, R.; SCHOOP, R. A solution to holonic control systems. **Proceedings of 8th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)**. v. 2, p. 489–498, 2001.

COLOMBO, A. W. et al. A collaborative automation approach to distributed production systems. **2nd IEEE International Conference on Industrial Informatics (INDIN)**. p. 27–32, 2004.

DA SILVA, R. M. et al. A service-oriented and holonic control architecture to the reconfiguration of dispersed manufacturing systems. In: **Technological Innovation for Collective Awareness Systems**. Berlin: Springer, 2014. v. 423, p. 111–118.

DA SILVA, R. M. et al. A method to design a manufacturing control system considering flexible reconfiguration. **Industrial Informatics, 2014. INDIN 2014. 12th IEEE International Conference on**. p. 82–87, 2014. DOI: 10.1109/INDIN.2014.6945488.

DA SILVA, R. M. et al. Control architecture and design method of reconfigurable manufacturing systems. **Control Engineering Practice**, Elsevier, v. 49, p. 87–100, 2016. DOI: 10.1016/j.conengprac.2016.01.009.

DA SILVA, R. M.; MIYAGI, P. E.; FILHO, D. J. S. Design of active holonic fault-tolerant control systems. **IFIP Advances in Information and Communication Technology**, Springer, v. 349, p. 367–374, 2011.

DA SILVA, R. M. et al. Modeling of mechanisms for reconfigurable and distributed manufacturing control system. **IFIP Advances in Information and Communication Technology**, Springer, v. 450, p. 93–100, 2015.

DELOACH, S. A.; GARCIA-OJEDA, J. C. The O-MaSE methodology. In: **Handbook on Agent-Oriented Design Processes. Handbook on Agent-Oriented Design Processes**. Berlin: Springer, 2014. p. 253–285.

DILTS, D.; BOYD, N.; WHORMS, H. The evolution of control architectures for automated manufacturing systems. **Journal of Manufacturing Systems**, Elsevier, v. 10, n. 1, p. 79–93, 1991.

ERL, T. **Service-oriented architecture**. New York: Prentice Hall, 2005. v. 8.

FARAHBOD, R. **Extending and Refining an Abstract Operational Semantics of the Web Services Architecture for the Business Process Execution Language**. 2001. Dissertação (Mestrado) — Sharif University of Technology, 2001.

FATTORI, C. et al. Description of productive processes in a collaborative environment. **37th IEEE Annual Conference on Industrial Electronics Society, IECON**. p. 379–384, 2011.

FATTORI, C. C. et al. Service composition modeling using interpreted Petri net for system integration. **IEEE International Conference on Mechatronics**. p. 696–701, 2011.

FELDMANN, K. et al. Specification, design, and implementation of logic controllers based on colored petri net models and the standard IEC61131. Part I. specification and design. **IEEE Transactions on Control Systems Technology**, IEEE, v. 7, n. 6, p. 657–665, 1999.

FERBER, J. **Multi-agent systems: an introduction to distributed artificial intelligence**. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 1999.

FERREIRA, A. B. H. Dicionário Aurelio online. Disponível em <http://www.dicionariodoaurelio.com>. Acessado em 10-02-2016. 2016.

FEURZEIG, W.; PAPERT, S. A.; LAWLER, B. Programming-languages as a conceptual framework for teaching mathematics. **Interactive Learning Environments**, Taylor & Francis, v. 19, n. 5, p. 487–501, 2011.

FIPA. Foundation for Intelligent Physical Agents. <http://www.fipa.org>, 2002.

FUGITA, H. S.; HIRAMA, K. **SOA: Modelagem, análise e design**. Campus, Rio de Janeiro: Elsevier, 2012.

GIRET, A.; BOTTI, V. Holons and agents. **Journal of Intelligent Manufacturing**, Springer, v. 15, n. 5, p. 645–659, 2004.

GIRET, A.; BOTTI, V. Engineering holonic manufacturing systems. **Computers in Industry**, Elsevier, v. 60, n. 6, p. 428–440, 2009.

GIRET, A.; TRENTESAUX, D. Software engineering methods for intelligent manufacturing systems: a comparative survey. **Lecture Notes in Computer Science**, v. 9266, p. 11–21, 2015.

GOMES, L. et al. The Input-Output Place-Transition Petri net class and associated tools. **5th IEEE International Conference on Industrial Informatics (INDIN)**. p. 27–32, 2007.

GOMES, L. et al. From Petri net models to VHDL implementation of digital controllers. **33rd Annual Conference of the IEEE Industrial Electronics Society (IECON)**. p. 94–99, 2007.

GROBA, C. et al. A service-oriented approach for increasing flexibility in manufacturing. **International Workshop on Factory Communication Systems**. p. 415 –422, may 2008. DOI: 10.1109/WFCS.2008.4638735.

GROOVER, M. P. **Automation, production systems, and computer-integrated manufacturing**. New Jersey: Prentice Hall Press, 2007.

GROOVER, M. P. **Automação industrial e sistemas de manufatura, 3.ed.** São Paulo: Pearson Prentice Hall, 2011.

HAN, R. et al. A Petri net theory-based method for modeling web service-based systems. **International Conference on Wireless Communications, Networking and Mobile Computing**. p. 1–7, Oct. 2008. DOI: 10.1109/WiCom.2008.2839.

HARRISON, R.; COLOMBO, A. W. Collaborative automation from rigid coupling towards dynamic reconfigurable production systems. **World Congress**. v. 16, n. 1, p. 1570–1570, 2005.

HASEGAWA, K. et al. On resource arc for Petri net modelling of complex resource sharing system. **Journal of Intelligent and Robotic Systems**, Springer, v. 26, n. 3-4, p. 423–437, 1999.

HAYES-ROTH, B. A blackboard architecture for control. **Artificial intelligence**, Elsevier, v. 26, n. 3, p. 251–321, 1985.

HSIEH, F.-S. Dynamic composition of holonic processes to satisfy timing constraints with minimal costs. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 22, n. 7, p. 1117–1126, 2009.

HSIEH, F.-S. Design of reconfiguration mechanism for holonic manufacturing systems based on formal models. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 23, n. 7, p. 1187–1199, 2010.

HSIEH, F.-S.; CHIANG, C. Y. Collaborative composition of processes in holonic manufacturing systems. **Computers in Industry**, Elsevier, v. 62, n. 1, p. 51–64, 2011.

HSIEH, F.-S.; LIN, J.-B. A self-adaptation scheme for workflow management in multi-agent systems. **Journal of Intelligent Manufacturing**, Springer, p. 1–18, 2013.

JENNINGS, N. R. An agent-based approach for building complex software systems. **Communications of the ACM**, ACM, v. 44, n. 4, p. 35–41, 2001.

JENNINGS, N. R.; BUSSMANN, S. Agent-based control systems. **IEEE Control Systems**, v. 23, n. 3, p. 61–74, 2003.

JENNINGS, N. R.; WOOLDRIDGE, M. J. Applications of intelligent agents. Berlin, Springer, 1998.

JOHN, K.-H.; TIEGELKAMP, M. **IEC 61131-3: programming industrial automation systems: concepts and programming languages, requirements for programming systems, decision-making aids**. Berlin: Springer Science & Business Media, 2010.

JOHNSON, R. E.; RUSSO, V. F. Reusing object-oriented designs. Citeseer, 1991.

JUNQUEIRA, F.; MIYAGI, P. A new method for the hierarchical modelling of productive systems. **Information Technology For Balanced Manufacturing Systems**, Springer, p. 479–488, 2006.

JUNQUEIRA, F.; VILLANI, E.; MIYAGI, P. E. A platform for distributed modeling and simulation of productive systems based on Petri nets and object-oriented paradigm. **International Conference on Emerging Technologies and Factory Automation**. 2005.

KANESHIRO, P. J. et al. Modeling of collision resolution algorithm in Lonworks networks. **Proceedings of ASME International Mechanical Engineering Congress and Exposition**. v. 9, p. 743–749, nov 2008.

KIEPUSZEWSKI, B.; HOFSTED, A.; AALST, W. Van der. Fundamentals of control flow in workflows. **Acta Informatica**, v. 39, n. 3, p. 143–209, 2003. DOI: 10.1007/s00236-002-0105-4.

KLINGSTAM, P.; GULLANDER, P. Overview of simulation tools for computer-aided production engineering. **Computers in Industry**, Elsevier, v. 38, n. 2, p. 173–186, 1999.

KOESTLER, A. The ghost in the machine. Macmillan, 1968.

KOLBERG, D.; ZÜHLKE, D. Lean automation enabled by Industry 4.0 technologies. **INCOM 15th IFAC/IEEE/IFIP/IFORS Symposium on Information Control Problems in Manufacturing**. p. 1919–1924, 2015.

KOREN, Y. Reconfigurable manufacturing system. In: **CIRP Encyclopedia of Production Engineering**. Berlin: Springer, 2014. p. 1035–1039.

KOREN, Y.; SHPITALNI, M. Design of reconfigurable manufacturing systems. **Journal of Manufacturing Systems**, Elsevier, v. 29, n. 1, p. 130–141, 2010.

KOSANKE, K.; VERNADAT, F.; ZELM, M. CIMOSA: enterprise engineering and integration. **Computers in Industry**, Elsevier, v. 40, n. 2, p. 83–97, 1999.

KUIPERS, B. **Qualitative reasoning: modeling and simulation with incomplete knowledge**. [S.l.]: the MIT press, 1994.

LEE, J.-S.; ZHOU, M.-C.; HSU, P.-L. An application of Petri nets to supervisory control for human-computer interactive systems. **IEEE Transactions on Industrial Electronics**, v. 52, n. 5, p. 1220–1226, Oct. 2005. DOI: 10.1109/TIE.2005.855694.

LEITÃO, P. Agent-based distributed manufacturing control: a state-of-the-art survey. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 22, n. 7, p. 979–991, 2009.

LEITÃO, P.; COLOMBO, A. W. Petri net based methodology for the development of collaborative production systems. **IEEE Conference on Emerging Technologies and Factory Automation (ETFA)**. p. 819–826, 2006.

LEITÃO, P. et al. Engineering tools for the integration of service-oriented production systems. **Information Control Problems in Manufacturing**. v. 13, n. 1, p. 1772–1777, 2009.

LEITÃO, P.; RESTIVO, F. ADACOR: A holonic architecture for agile and adaptive manufacturing control. **Computers in Industry**, Elsevier, v. 57, n. 2, p. 121–130, 2006.

LEITÃO, P. et al. Intelligent products: the grace experience. **Control Engineering Practice**, Elsevier, v. 42, p. 95–105, 2015.

LEMAIGNAN, S. et al. MASON: A proposal for an ontology of manufacturing domain. **IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications**. p. 195–200, 2006.

LEPUSCHITZ, W. et al. A survey on standards and ontologies for process automation. In: **Industrial Applications of Holonic and Multi-Agent Systems**. Berlin: Springer, 2015. p. 22–32.

LI, B.-H. et al. Cloud manufacturing: a new service-oriented networked manufacturing model. **Computer Integrated Manufacturing Systems**, v. 16, n. 1, p. 1–7, 2010.

LI, Z.; ZHOU, M. Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems. **IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans**, v. 34, n. 1, p. 38–51, Jan. 2004. DOI: 10.1109/TSMCA.2003.820576.

LI, Z.; ZHOU, M. Control of elementary and dependent siphons in Petri nets and their application. **IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans**, v. 38, n. 1, p. 133–148, Jan. 2008. DOI: 10.1109/TSMCA.2007.909548.

LIRA, D. et al. Fault detection in flexible assembly systems using Petri net. **IEEE Latin America Transactions**, IEEE, v. 6, n. 7, p. 572–578, dec. 2008. DOI: 10.1109/TLA.2008.4917427.

LUKE, S. et al. Mason: a multiagent simulation environment. **Simulation**, Sage Publications, v. 81, n. 7, p. 517–527, 2005.

MAIONE, G.; NASO, D. A genetic approach for adaptive multiagent control in heterarchical manufacturing systems. **IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans**, IEEE, v. 33, n. 5, p. 573–588, 2003.

MAŘÍK, V.; LAŽANSKÝ, J. Industrial applications of agent technologies. **Control Engineering Practice**, Elsevier, v. 15, n. 11, p. 1364–1380, 2007.

MAŘÍK, V. et al. **Industrial applications of holonic and multi-agent systems**. Berlin: Springer, 2015. v. 9266.

MATURANA, F.; SHEN, W.; NORRIE, D. H. Metamorph: an adaptive agent-based architecture for intelligent manufacturing. **International Journal of Production Research**, Taylor & Francis, v. 37, n. 10, p. 2159–2173, 1999.

MEHRABI, M. G.; ULISOY, A. G.; KOREN, Y. Reconfigurable manufacturing systems and their enabling technologies. **International Journal of Manufacturing Technology and Management**, Inderscience, v. 1, n. 1, p. 114–131, 2000.

MELO, J. I. et al. Framework for collaborative manufacturing systems based in services. **Proceedings of the International Congress of Mechanical Engineering**. nov 2009.

MELO, J. I. et al. A procedure for modeling and analysis of service-oriented and distributed productive systems. **International Conference on Automation Science and Engineering**. p. 941–946, Aug. 2008. DOI: 10.1109/COASE.2008.4626561.

MELO, J. I.; MORALES, R. A. G.; MIYAGI, P. E. Supervisory system for hybrid productive systems based on Bayesian networks and OO-DPT nets. **International Conference on Emerging Technologies and Factory Automation**. p. 1108–1111, Sept. 2008. DOI: 10.1109/ETFA.2008.4638536.

MELZER, I. **Service-orientierte Architekturen mit Web Services: Konzepte-Standards-Praxis**. [S.l.]: Spektrum Akademischer Verlag, 2010.

MENDES, J. M. et al. Service-oriented agents for collaborative industrial automation and production systems. In: **Holonic and Multi-agent Systems for Manufacturing**. Berlin: Springer, 2009. p. 13–24.

MICK, R.; POLSONETTI, C. Collaborative automation: the platform for operational excellence. **White paper, ARC Advisory Group**, 2003.

MILLER, E. An introduction to the resource description framework. **Bulletin of the American Society for Information Science and Technology**, Wiley Online Library, v. 25, n. 1, p. 15–19, 1998.

MINAR, N. et al. The swarm simulation system: a toolkit for building multi-agent simulations. 1996.

MIYAGI, P.; RIASCOS, L. Modeling and analysis of fault-tolerant systems for machining operations based on Petri nets. **Control Engineering Practice**, Elsevier, v. 14, n. 4, p. 397–408, 2006.

MIYAGI, P. E. **Controle Programável - Fundamentos do controle de sistemas a eventos discretos**. São Paulo, SP: Editora Edgard Blucher LTDA, 1996.

MORALES, R. A. G.; MELO, J. I.; MIYAGI, P. E. Diagnosis and treatment of faults in productive systems based on Bayesian networks and Petri net. **International Conference on Automation Science and Engineering**. p. 357–362, Sept. 2007. DOI: 10.1109/COASE.2007.4341823.

MOREL, G. et al. Manufacturing plant control challenges and issues. **Control Engineering Practice**, Elsevier, v. 15, n. 11, p. 1321–1331, 2007.

MOSLEMIPOUR, G.; LEE, T. S.; RILLING, D. A review of intelligent approaches for designing dynamic and robust layouts in flexible manufacturing systems. **The International Journal of Advanced Manufacturing Technology**, Springer, v. 60, n. 1-4, p. 11–27, 2012.

MOUTINHO, F.; GOMES, L. Asynchronous-channels and time-domains extending Petri nets for GALS systems. In: **Technological Innovation for Value Creation**. [S.l.]: Springer, 2012. p. 143–150.

MPS. The Modular Production System. Disponível em <http://www.festo-didactic.com/int-en/learning-systems/mps-the-modular-production-system>. Acessado em 15-06-2015. 2015.

MURATA, T. Petri nets: properties, analysis and applications. **Proceedings of IEEE**, v. 77, n. 4, p. 541–580, 1989.

NAGORNY, K.; COLOMBO, A. W.; SCHMIDTMANN, U. A service-and multi-agent-oriented manufacturing automation architecture: an IEC 62264 level 2 compliant implementation. **Computers in Industry**, Elsevier, 2012.

NASSAR, M. G. V. et al. Modeling and analysing of the material entry flow system in a picklingline process using Petri nets. **ABCM Symposium Series in Mechatronics**. v. 3, p. 444–453, 2008.

NORTH, M. J. et al. A declarative model assembly infrastructure for verification and validation. **Advancing Social Simulation: The First World Congress**. p. 129–140, 2007.

OBITKO, M.; MAŘÍK, V. Adding OWL semantics to ontologies used in multi-agent systems for manufacturing. In: **Holonic and Multi-Agent Systems for Manufacturing**. Berlin: Springer, 2003. p. 189–200.

OBITKO, M. et al. Applications of semantics in agent-based manufacturing system. **Informatica (Slovenia)**, v. 34, n. 3, p. 315–330, 2010.

OUNNAR, F. et al. Integration of a flat holonic form in an HLA environment. **Journal of Intelligent Manufacturing**, Springer, v. 20, n. 1, p. 91–111, 2009.

PACH, C. et al. An effective potential field approach to FMS holonic heterarchical control. **Control Engineering Practice**, Elsevier, v. 20, n. 12, p. 1293–1309, 2012.

PAIS, R.; BARROS, J. P.; GOMES, L. From SysML State Machines to Petri Nets Using ATL Transformations. In: **Technological Innovation for Collective Awareness Systems**. [S.I.]: Springer, 2014. p. 227–236.

PARUNAK, H. V. D.; BAKER, A. D.; CLARK, S. J. The AARIA agent architecture: from manufacturing requirements to agent-based system design. **Integrated Computer Aided Engineering**, IOS Press, v. 8, n. 1, p. 45–58, 2001.

PĚCHOUČEK, M.; MAŘÍK, V. Industrial deployment of multi-agent technologies: review and selected case studies. **Autonomous Agents and Multi-Agent Systems**, Springer, v. 17, n. 3, p. 397–431, 2008.

PELTZ, C. Web services orchestration and choreography. **Computer**, IEEE, v. 36, n. 10, p. 46–52, 2003.

PEREIRA, F. et al. Web based IOPT Petri net editor with an extensible plugin architecture to support generic net operations. **38th IEEE Annual Conference on Industrial Electronics Society (IECON)**. p. 6151–6156, 2012.

PÉREZ, J.; ARENAS, M.; GUTIERREZ, C. Semantics and complexity of SPARQL. **International Semantic Web Conference**. v. 4273, p. 30–43, 2006.

PORCARELLI, S. et al. A framework for reconfiguration-based fault-tolerance in distributed systems. In: **Architecting Dependable Systems II. Architecting Dependable Systems II**. Berlin: Springer, 2004. p. 167–190.

POSLAD, S.; BUCKLE, P.; HADINGHAM, R. The fipa-os agent platform: open source for open standards. **Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents**. v. 355, p. 368, 2000.

PUJO, P.; BROISSIN, N.; OUNNAR, F. PROSIS: an isoarchic structure for HMS control. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 22, n. 7, p. 1034–1045, 2009.

PUTNIK, G. et al. Scalability in manufacturing systems design and operation: state-of-the-art and future developments roadmap. **CIRP Annals-Manufacturing Technology**, Elsevier, v. 62, n. 2, p. 751–774, 2013.

QUINTANILLA, F. G. et al. Implementation of a process orchestration model in a service oriented holonic manufacturing system. **Information Control Problems in Manufacturing**. v. 15, n. 1, p. 1167–1172, 2015.

RAMADGE, P. J.; WONHAM, W. M. The control of discrete event systems. **Proceedings of the IEEE**, IEEE, v. 77, n. 1, p. 81–98, 1989.

REISIG, W. Petri nets: an introduction. **Springer Verlag, New York**, 1985.

RIBEIRO, J. et al. An ecore based Petri net type definition for PNML IOPT models. **Industrial Informatics (INDIN), 2011 9th IEEE International Conference on**. p. 777–782, 2011.

ROBINSON, P. J. Hierarchical object oriented design. **Conference proceedings on Data Management 91**. p. 87–92, 1992.

RUSSELL, S. J. et al. **Artificial intelligence: a modern approach**. Englewood Cliffs, NJ: Prentice Hall, 1995. v. 2.

SALLEZ, Y.; BERGER, T.; TRENTESAUX, D. A stigmergic approach for dynamic routing of active products in FMS. **Computers in Industry**, v. 60, p. 204–216, 2009.

SAMPATH, M. et al. Failure diagnosis using discrete-event models. **IEEE Transactions on Control Systems Technology**, IEEE, v. 4, n. 2, p. 105–124, 1996.

SCHMITT, M.; ORFGEN, M.; ZÜHLKE, D. Dynamic reconfiguration of intelligent field devices by using modular software applications. **Information Control Problems in Manufacturing**. v. 15, n. 1, p. 595–600, 2015.

SHEN, W. et al. An agent-based service-oriented integration architecture for collaborative intelligent manufacturing. **Robotics and Computer-Integrated Manufacturing**, Elsevier, v. 23, n. 3, p. 315–325, 2007.

SMITH, R. The contract net protocol: high-level communication and control in a distributed problem solver. **IEEE Transactions on Computers**, IEEE, v. 100, n. 12, p. 1104–1113, 1980.

SQUILLANTE, R. et al. Mathematical method for modeling and validating of safety instrumented system designed according to IEC61508 and IEC61511. In: **ABCM Symposium Series in Mechatronics**. Rio de Janeiro, RJ: ABCM, 2012. 2, p. 484–493.

STRASSER, T. et al. A research roadmap for model-driven design of embedded systems for automation components. **7th IEEE International Conference on Industrial Informatics (INDIN)**. p. 564–569, 2009.

TISUE, S.; WILENSKY, U. Netlogo: a simple environment for modeling complexity. **International Conference on Complex Systems (ICCS)**. p. 16–21, 2004.

TOBIAS, R.; HOFMANN, C. Evaluation of free java-libraries for social-scientific agent based simulation. **Journal of Artificial Societies and Social Simulation**, v. 7, n. 1, 2004.

TOMMILA, T. et al. Next generation of industrial automation concepts and architecture of a component-based control system. **VTT Tiedotteita. Research Notes**, VALTION TEKNIILLINEN TUTKIMUSKESKUS, ESPOO, v. 2303, 2005.

TRAN, Q.-N. N.; LOW, G. MOBMAS: a methodology for ontology-based multi-agent systems development. **Information and Software Technology**, Elsevier, v. 50, n. 7, p. 697–722, 2008.

TSENG, M. M.; HU, S. J. Mass customization. In: **CIRP Encyclopedia of Production Engineering**. Berlin: Springer, 2014. p. 836–843.

VALCKENAERS, P. et al. Results of the holonic control system benchmark at ku leuven. **Proceedings of the Fourth International Conference on Computer Integrated Manufacturing and Automation Technology**. p. 128–133, 1994.

Van Brussel, H. et al. Reference architecture for holonic manufacturing systems: PROSA. **Computers in Industry**, Elsevier, v. 37, n. 3, p. 255–274, 1998.

VILLANI, E.; MIYAGI, P. E.; VALETTE, R. **Modelling and analysis of hybrid supervisory systems: a Petri net approach**. Berlin: Springer Verlag, 2007. v. 1430.

VRBA, P. Java-based agent platform evaluation. **Holonic and Multi-Agent Systems for Manufacturing**, Springer, p. 1086–1087, 2003.

VRBA, P. Simulation in agent-based control systems: mast case study. **International Journal of Manufacturing Technology and Management**, Inderscience Publishers, v. 8, n. 1-3, p. 175–187, 2006.

VRBA, P. et al. Rockwell automation's holonic and multiagent control systems compendium. **IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews**, IEEE, v. 41, n. 1, p. 14–30, 2011.

VYATKIN, V. V.; CHRISTENSEN, J. H.; LASTRA, J. M. OOONEIDA: An open, object-oriented knowledge economy for intelligent industrial automation. **IEEE Transactions on Industrial Informatics**, IEEE, v. 1, n. 1, p. 4–17, 2005.

WANG, W.; KOREN, Y. Scalability planning for reconfigurable manufacturing systems. **Journal of Manufacturing Systems**, Elsevier, v. 31, n. 2, p. 83–91, 2012.

WILENSKY, U. Netlogo traffic grid model. **Center for Connected Learning and Computer-Based Modeling, Northwestern U., Evanston, IL**, 2003.

XU, X. et al. A novel modeling design method for automated storage and retrieval system based on Petri nets. **Proceedings of the International Conference on Automation and Logistics**. p. 2046–2051, aug 2007.

YOO, T.; JEONG, B.; CHO, H. A Petri nets based functional validation for services composition. **Expert Systems with Applications**, v. 37, p. 3768–3776, 2010. DOI: 10.1016/j.eswa.2009.11.046.

ZHANG, X. et al. Design and implementation of a real-time holonic control system for manufacturing. **Information Sciences**, Elsevier, v. 127, n. 1, p. 23–44, 2000.

ZHANG, Y.; JIANG, J. Bibliographical review on reconfigurable fault-tolerant control systems. **Annual Reviews in Control**, Elsevier, v. 32, n. 2, p. 229–252, 2008.

ZHENG, L.; NAKAGAWA, H. OPC (OLE for Process Control) specification and its developments. **SICE 2002. Proceedings of the 41st SICE Annual Conference**. v. 2, p. 917–920, 2002.

ZURAWSKI, R.; ZHOU, M. Petri nets and industrial applications: a tutorial. **IEEE Transactions on Industrial Eletronics**, v. 41, n. 6, p. 567–583, dec 1994.

Apêndice A – Código para implementação do modelo Netlogo

Na Listagem 11 é apresentado o código completo para implementação do modelo desenvolvido no Netlogo (WILENSKY, 2003) versão 5.3.1. Assim, o leitor pode compreender melhor a solução proposta e facilita-se o aprimoramento/ampliação deste trabalho ou o desenvolvimento de outras aplicações.

Listagem 11 - Código completo para implementação do modelo Netlogo apresentado no Capítulo 5.

```
;; Modelo desenvolvido por Robson Marinho da Silva
;;Este modelo foi criado usando a versão 5.3.1 (2016) do Netlogo.
;;Para usar em outras versões deve ser verificado a compatibilidade.
;;Para executar o programa além desse código, devem ser criados na área de
;;trabalho elementos do Netlogo: ‘‘monitors’’, ‘‘bottoms’’, ‘‘plot’’,
;; ‘‘switchers’’ e ‘‘sliders’’, conforme as Figuras 47, 63, 65 e 66.
;; Se for usar este modelo em publicação deve ser citado o artigo:
;; da Silva , R. M.; Junqueira , F.; Santos Filho , D. J.; Miyagi , P. E.
;; Control architecture and design method of reconfigurable
;; manufacturing systems. Control Engineering Practice ,
;; v. 49, p. 87-100, 2016. DOI:10.1016/j.conengprac.2016.01.009.
```

```

; :::::::::::::: CLASSES ::::::::::::::
breed [ PrH-pallet ] breed [ PrH-cylinderBody ] breed [ PrH-pistonAp ]
breed [ PrH-pistonBp ] breed [ PrH-spring ] breed [ PrH-cover ]
breed [ TH-CustOrder]

; :::::::::::::: VARIÁVEIS ::::::::::::::
;;utilizadas por todos os agentes;;

globals

[ In-Free-BufferDWS  IN-free-PrH-cylinderBody  new_directions
distance-to-jump  FinalProduct-outputTotal
FinalProduct-outputPrH_bcb_ap  FinalProduct-outputPrH_rcb_ap
FinalProduct-outputPrH_acb_ap  FinalProduct-outputPrH_bcb_bp
FinalProduct-outputPrH_rcb_bp  FinalProduct-outputPrH_acb_bp
FinalProduct-outputPrH_bcb  FinalProduct-outputPrH_rcb
FinalProduct-outputPrH_acb  FinalProduct-output-bad
wip  mlt-aux  avg-mlt  targetColorPrH-pallet
binApFree  binBpFree  binSpFree  binPrH-coverFree
use-TeWS2-Automatic

TimeDWS  TimeTeWS1  TimeTeWS2  TimeIWS  TimeSWS  TimeCNC_A

```

```

TimeCNC_B   TimeRMT_acb   TimeRMT_rcb   TimeAWS   TimeQCWS
TimeTePalWS   TimeFinal
pistonColor  ap  bp  none-piston  cylBodyColor  quaCusOrd
cusOrdNum  quaRcb  quaBcb  quaAcb  ThCoList
totalAcbAp  totalBcbAp  totalRcbAp  totalAcbBp  totalBcbBp  totalRcbbp
]
;;usada somente para os agentes patches;;;
patches-own
[ directions  velocity  processing-time
is-processing?  failure  output ID  bifurcation ]
;;usadas somente para os agentes PrH-pallets;;;
PrH-pallet-own
[ birth-time  death-time  priority  time-to-duedate  productionPlan
processing  time-of-processing  QC-check ]
;;usada somente para os agentes PrH;;
PrH-cylinderBody-own
[ birth-time  productionPlan  processing  time-of-processing ]
PrH-pistonAp-own
[ birth-time  productionPlan  processing  time-of-processing ]
PrH-pistonBp-own
[ birth-time  productionPlan  processing  time-of-processing ]
PrH-spring-own
[ birth-time  productionPlan  processing  time-of-processing ]
PrH-cover-own
[ birth-time  productionPlan  processing  time-of-processing ]
;; usadas somente para agentes TH
TH-CustOrder-own
[ dueDate  pistonTH-CO  birth-time  thData ]
;;;;;;;;; setup ;;;;;;;;;;
to setup
clear-all
initGlobals  setupCycleRoute  setupWS  setupPWSRoutes  setupAWSRoutes
setupTEWS2Routes  setupSWSRoutes  setupOUTWSRoutes  setupbuffers
setupBuffersRoutes  setupCorners  setupLabels  setupSuH-Agenda
reset-ticks
end
;;;;;;;;; SuHAgenda ;;;;;;;;;;
to setupSuH-Agenda
ask patch -11 -2
[ set plabel "SuH-SupAgenda"  set plabel-color black
set totalAcbAp 0  set totalBcbAp 0  set totalRcbAp 0
set totalAcbBp 0  set totalBcbBp 0  set totalRcbBp 0
]

```

```

end
;;;;;;;;; InitGlobals ;;;;;;;;;;
to initGlobals
set FinalProduct-outputTotal 0 set FinalProduct-outputPrH_bcb_ap 0
set FinalProduct-outputPrH_rcb_ap 0 set FinalProduct-outputPrH_acb_ap 0
set FinalProduct-outputPrH_bcb_bp 0 set FinalProduct-outputPrH_rcb_bp 0
set FinalProduct-outputPrH_acb_bp 0 set FinalProduct-outputPrH_bcb 0
set FinalProduct-outputPrH_rcb 0 set FinalProduct-outputPrH_acb 0
set avg-mlt 0 set mlt-aux 0 set FinalProduct-output-bad 0
ask patches [ set pcolor white ]
set cylBodyColor [-1 -1 -1 -1] set pistonColor [-1 -1 -1 -1]
set quaRcb [0 0] set quaBcb [0 0] set quaAcB [0 0]
set ThCoList [] set cusOrdNum 0
end
;;;;;;;;; Desenvolvimento de do TH-TrWS - rotas ;;;;;;;;;;
;;;;;;;;; horizontal pycor=0
to setupCycleRoute
ask patches [ if pxcor >= -9 and pxcor <= -8 and pycor = 0
[ set pcolor brown set directions [-1 90 -1 -1]
set velocity 1 set is-processing? FALSE
set failure FALSE set bifurcation FALSE ] ]
ask patch -10 0 [ set pcolor brown set directions [-1 90 -1 -1]
set velocity 1 set is-processing? FALSE set failure FALSE
set bifurcation FALSE set id "corner0"]
ask patch -7 0 [ set pcolor brown set directions [-1 90 -1 -1]
set velocity 1 set is-processing? FALSE set failure FALSE
set bifurcation FALSE ]
ask patch -6 0 [ set pcolor brown set directions [-1 90 -1 -1]
set velocity 1 set is-processing? FALSE set failure FALSE
set bifurcation FALSE ]
ask patches [ if pxcor <= 2 and pxcor >= 1 and pycor = 0
[ set pcolor brown set directions [-1 90 -1 -1]
set velocity 1 set is-processing? FALSE
set failure FALSE set bifurcation FALSE ] ]
ask patches [ if pxcor >= -4 and pxcor <= -3 and pycor = 0
[ set pcolor brown set directions [-1 90 -1 -1]
set velocity 1 set is-processing? FALSE
set failure FALSE set bifurcation FALSE ] ]
ask patch -2 0 [ set pcolor brown set directions [0 90 180 -1]
set velocity 1 set is-processing? FALSE set failure FALSE
set bifurcation TRUE set ID "InTeWS" ]
ask patch -1 0 [
set pcolor brown set directions [-1 90 -1 -1] set velocity 1

```

```

set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
ask patches [ if pxcor >= 4 and pxcor <= 6 and pycor = 0
[ set pcolor brown    set directions [-1 90 -1 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE
set bifurcation FALSE ] ]
ask patches [ if pxcor >= 8 and pxcor <= 9 and pycor = 0
[ set pcolor brown    set directions [-1 90 -1 -1]
set velocity 1    set is-processing? FALSE    set failure FALSE
set bifurcation FALSE ] ]
;;;;;;;;; Vertical Pxcor =10 ;;;;;;;;;;
ask patches
[ if pxcor = 10 and pycor >= 1 and pycor <= 3
[ set pcolor brown    set directions [0 -1 -1 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ] ]
;;;;;;;;; Horizontal pycor=4 ;;;;;;;;;;
ask patches[ if pxcor >= -8 and pxcor <= 8 and pycor = 4
[ set pcolor brown    set directions [-1 -1 -1 270]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ] ]
;;;;;;;;; Vertical pxcor=-10 ;;;;;;;;;;
ask patches [ if pxcor = -10 and pycor >= 2 and pycor <= 3 [
set pcolor brown    set directions [-1 -1 180 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
]
end
;;;;;;;;; Criação de WS ;;;;;;;;;;
to setupWS
ask patch -5 1
[ set pcolor yellow    set directions [-1 -1 180 -1]    set velocity 1
set processing-time 20    set is-processing? FALSE    set failure FALSE
set ID "TePalWS"    set bifurcation FALSE    set TimeTePalWS 0 ]
ask patch 0 2 [
set pcolor yellow    set directions [-1 90 -1 -1]    set velocity 1
set processing-time 20    set is-processing? FALSE    set failure FALSE
set ID "TeWS1"    set bifurcation FALSE    set TimeTeWS1 0 ]
;;;;;;;;;Exemplo no texto – Listagem 1 ;;;;
ask patch -5 0 ;;Define a coordenada em que o agente é criado.
[ set pcolor yellow ;;Define a cor do patch.
set directions [-1 90 -1 -1] ;;Define uma lista com possíveis direções em
relação ao eixo horizontal
;;e que é modificada de acordo com o movimento na TrWS.
set velocity 1 ;;Define a velocidade, relacionando o número de patch que
cada agente PrH irá se mover em cada tick.

```

```

set processing-time 20 ;; Define um tempo de processamento considerado para
    cada WS executar suas operações no PrH.
set is-processing? FALSE ;; Se valor é "TRUE" significa que a WS está
    executando operações em um PrH,
    ;; caso contrário o valor é "FALSE".
set failure FALSE ;; Se valor é "TRUE" significa que a WS está em estado de
    falha, caso contrário o valor é "FALSE".
set ID "DWS" ;; Define uma identificação para o agente.
set bifurcation FALSE ;; Define se o patch está sobre uma bifurcação.
set timeDWS 0 ;; Define o valor do instante de tempo como zero para
    contabilizar o tempo do agente.
]
;;;;;;;;;; definição de agentes patches
ask patch 7 0 [
set pcolor yellow    set directions [-1 90 180 -1]    set velocity 1
set processing-time 20    set is-processing? FALSE    set failure FALSE
set ID "SWS"    set bifurcation TRUE    set TimeSWS 0
]
ask patch 7 6 [
set pcolor yellow    set directions [-1 -1 -1 270]    set velocity 1
set processing-time 20    set is-processing? FALSE    set failure FALSE
set ID "CNC A"    set bifurcation FALSE    set TimeCNC_A 0
]
ask patch 7 10 [
set pcolor yellow    set directions [-1 -1 -1 270]    set velocity 1
set processing-time 20    set is-processing? FALSE    set failure FALSE
set ID "RMT rcb"    set bifurcation FALSE    set TimeRMT_rcb 0
]
ask patch 3 6 [
set pcolor yellow    set directions [-1 -1 -1 270]    set velocity 1
set processing-time 20    set is-processing? FALSE    set failure FALSE
set ID "CNC B"    set bifurcation FALSE    set TimeCNC_B 0
]
ask patch 3 10 [
set pcolor yellow    set directions [-1 -1 -1 270]    set velocity 1
set processing-time 20    set is-processing? FALSE    set failure FALSE
set ID "RMT acb"    set bifurcation FALSE    set TimeRMT_acb 0
]
ask patch 0 -2 [
set pcolor yellow    set directions [-1 90 -1 -1]    set velocity 1
set processing-time 20    set is-processing? FALSE    set failure FALSE
set ID "TeWS2"    set bifurcation FALSE    set TimeTeWS2 0
]

```

```

ask patch 3 0 [
set pcolor yellow    set directions [-1 90 -1 -1]    set velocity 1
set processing-time 20    set is-processing? FALSE    set failure FALSE
set ID "IWS"    set bifurcation FALSE    set TimeIWS 0
]
ask patch 0 6 [
set pcolor yellow    set directions [-1 -1 -1 270]    set velocity 1
set processing-time 50    set is-processing? FALSE    set failure FALSE
set ID "AWS"    set bifurcation FALSE    set TimeAWS 0
]
ask patch -9 4[
set pcolor yellow    set directions [-1 -1 -1 270]    set velocity 1
set processing-time 20    set is-processing? FALSE    set failure FALSE
set ID "QCWS"    set bifurcation FALSE    set TimeQCWS 0
]
end
;;;;;;;;;; rotas da TeWS ;;;;;;;;;;
to setupTeWS2Routes
ask patch 2 1
[ set pcolor brown    set directions [-1 -1 180 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
]
ask patch 2 2
[ set pcolor brown    set directions [-1 -1 180 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
]
ask patch 1 2
[ set pcolor brown    set directions [-1 90 -1 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
]
ask patch 0 0
[ set pcolor brown    set directions [-1 90 -1 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
]
ask patch -2 1
[ set pcolor brown    set directions [0 -1 -1 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
]
ask patch -2 2
[ set pcolor brown    set directions [-1 90 -1 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
]
ask patch -1 2

```

```

[ set pcolor brown    set directions [-1 90 -1 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
]
ask patch -2 -1 [
set pcolor brown    set directions [-1 -1 180 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
]
ask patch -2 -2 [
set pcolor brown    set directions [-1 90 -1 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
]
ask patch -1 -2 [
set pcolor brown    set directions [-1 90 -1 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
]

ask patch 1 -2 [
set pcolor brown    set directions [-1 90 -1 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
]
ask patch 2 -2 [
set pcolor brown    set directions [0 -1 -1 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
set ID "TeWSCorner"
]
ask patch 2 -1 [
set pcolor brown    set directions [0 -1 -1 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
]
end
;;;;;;;;;SWS;;;;;;;;;
to setupSWSRoutes
ask patch 7 -1
[ set pcolor brown    set directions [-1 -1 180 -1]    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
]
ask patch 7 -2
[ set pcolor brown    set directions [-1 -1 180 -1]    set velocity 0.6
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
]
end
;;;;;;;;;AWS;;;;;;;;;
to setupAWSRoutes

```

```

ask patch -8 5
[ set pcolor brown      set directions [-1 -1 180 -1]      set velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch -1 6
[ set pcolor brown      set directions [-1 -1 -1 270]      set velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE
set processing-time 2      set ID "BinApExit" ]
ask patch -3 6
[ set pcolor brown      set directions [-1 -1 -1 270]      set velocity 1
set is-processing? FALSE      set processing-time 2      set failure FALSE
set bifurcation FALSE      set ID "BinBpExit" ]
ask patch -5 6
[ set pcolor brown      set directions [-1 -1 -1 270]      set velocity 1
set is-processing? FALSE      set processing-time 2      set failure FALSE
set bifurcation FALSE      set ID "BinSpExit" ]
ask patch -7 6
[ set pcolor brown      set directions [-1 -1 -1 270]      set velocity 1
set is-processing? FALSE      set processing-time 2      set failure FALSE
set bifurcation FALSE      set ID "BinPrH-coverExit" ]
ask patch -8 6
[ set pcolor brown      set directions [-1 -1 180 -1]      set velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE
set ID "AwsExit"
]
ask patches
[ if (pxcor = -2 or pxcor = -4 or pxcor = -6) and pycor = 6
[ set pcolor brown      set directions [-1 -1 -1 270]      set velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
]
ask patch 0 5
[ set pcolor brown      set directions [0 -1 -1 -1]      set velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE
]
ask patch 0 4
[ set pcolor brown      set directions [0 -1 -1 270]      set velocity 1
set processing-time 2      set is-processing? FALSE      set failure FALSE
set bifurcation TRUE      set ID "InAWS"
]
;;;;;;;;;;;; buffers
ask patches [ if pxcor = -1 and pycor <= 9 and pycor >= 7
[ set pcolor brown      set directions [-1 -1 180 -1]      set velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE
] ]

```



```

ask patches [ if pxcor = -3 and pycor <= 9 and pycor >= 7
[ set pcolor brown      set directions [-1 -1 180 -1]      set velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE
] ]
ask patches [ if pxcor = -5 and pycor <= 9 and pycor >= 7
[ set pcolor brown      set directions [-1 -1 180 -1]      set velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE
]
]
ask patches [ if pxcor = -7 and pycor <= 9 and pycor >= 7
[ set pcolor brown      set directions [-1 -1 180 -1]      set velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE
]
]
end
;;;;;;;;;PWS;;;;;;;;;
to setupPWSRoutes
ask patch 6 5 [ set pcolor brown      set directions [-1 -1 180 -1]      set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation False ]
ask patch 9 4 [      set pcolor brown      set directions [0 -1 -1 270]      set
velocity 1
set is-processing? FALSE      set processing-time 1      set failure FALSE
set ID "InPWS"      set bifurcation TRUE ]
ask patch 9 5 [ set pcolor brown      set directions [0 -1 -1 -1]      set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch 9 6 [ set pcolor brown      set directions [0 -1 -1 270]      set
velocity 1
set is-processing? FALSE      set processing-time 5      set failure FALSE
set ID "InCNC"      set bifurcation TRUE ]
ask patch 9 7 [ set pcolor brown      set directions [0 -1 -1 -1]      set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch 9 8 [ set pcolor brown      set directions [0 -1 -1 -1]      set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch 9 9 [ set pcolor brown      set directions [0 -1 -1 -1]      set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patches [ if pxcor = 2 and pycor <= 10 and pycor >= 5
[ set pcolor brown      set directions [-1 -1 180 -1]      set velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ] ]

```

```

ask patch 4 10 [ set pcolor brown      set directions [-1 -1 -1 270]  set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch 5 10 [ set pcolor brown      set directions [-1 -1 180 270]  set
velocity 1
set processing-time 1      set is-processing? FALSE      set failure FALSE
set ID "InCNCB"      set bifurcation TRUE ]
ask patch 6 10 [ set pcolor brown      set directions [-1 -1 -1 270]  set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch 8 10 [ set pcolor brown      set directions [-1 -1 -1 270]  set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch 9 10 [ set pcolor brown      set directions [-1 -1 -1 270]  set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch 4 6 [ set pcolor brown      set directions [-1 -1 -1 270]  set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch 5 6 [ set pcolor brown      set directions [-1 -1 -1 270]  set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch 6 6 [ set pcolor brown      set directions [0 -1 180 270]  set
velocity 1
set is-processing? FALSE      set processing-time 1      set failure FALSE
set ID "InRMTacb"      set bifurcation TRUE ]
ask patch 8 6 [ set pcolor brown      set directions [-1 -1 -1 270]  set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch 5 7 [ set pcolor brown      set directions [-1 -1 180 -1]  set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch 5 8 [ set pcolor brown      set directions [-1 -1 180 -1]  set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch 5 9 [ set pcolor brown      set directions [-1 -1 180 -1]  set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch 6 7 [ set pcolor brown      set directions [0 -1 -1 -1]  set
velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE ]
ask patch 6 8 [ set pcolor brown      set directions [0 -1 -1 -1]  set
velocity 1

```

```

set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
ask patch 6 9 [ set pcolor brown    set directions [0 -1 -1 -1]    set
    velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
end
;;;;;;;;;;;;;OUTWS
to setupOUTWSRoutes
ask patch -11 1 [    set pcolor brown    set directions [-1 -1 -1 270]
    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
ask patch -12 1 [    set pcolor brown    set directions [-1 -1 -1 270]
    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
end
;;;;;;;;;;;;; Cylinder Body buffer ;;;;;;;;;;;
to setupBuffersRoutes
ask patch -5 -1 [    set pcolor brown    set directions [0 -1 -1 -1]    set
    velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
ask patch -5 -2 [    set pcolor brown    set directions [0 -1 -1 -1]    set
    velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
ask patch -5 -3 [    set pcolor brown    set directions [0 -1 -1 -1]    set
    velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
ask patch -5 -4 [    set pcolor brown    set directions [0 -1 -1 -1]    set
    velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
ask patch -5 -5 [    set pcolor brown    set directions [0 -1 -1 -1]    set
    velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
ask patch -5 -6 [    set pcolor brown    set directions [0 -1 -1 -1]    set
    velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
ask patch -5 -7 [    set pcolor brown    set directions [0 -1 -1 -1]    set
    velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
ask patch -5 -8 [    set pcolor Green    set directions [0 -1 -1 -1]    set
    velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
end
;;;;;;;;;;;;; PrH-cylinderBodys Buffer IN ;;;;;;;;;;;
to setupBuffers

```

```

ask patch -5 2 [    set pcolor Green    set directions [-1 -1 180 -1]
    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE ]
ask patch -10 1 [    set pcolor brown    set directions [-1 -1 180 270]
    set velocity 1
set is-processing? FALSE    set failure FALSE    set output TRUE    set
    bifurcation TRUE ]
;;;;;;;;;; Buffer Out ;;;;;;;;;;;
ask patch -13 1 [    set pcolor orange    set directions [-1 -1 -1 -1]
    set velocity 1
set is-processing? FALSE    set failure FALSE    set output TRUE    set ID
    "OUT"
set bifurcation FALSE ]
ask patch 7 -3 [    set pcolor orange    set directions [-1 -1 180 -1]
    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
    set ID "SWSOUT" ]
;;;;;;;;;;AWS Buffers ;;;;;;;;;;;
ask patch -1 10 [    set pcolor green    set directions [-1 -1 -1 -1]
    set velocity 1
set is-processing? FALSE    set failure FALSE    set output TRUE    set ID
    "BinBp"
set bifurcation FALSE ]
ask patch -3 10 [    set pcolor green    set directions [-1 -1 180 -1]
    set velocity 1
set is-processing? FALSE    set failure FALSE    set bifurcation FALSE
    set ID "BinAp" ]
ask patch -5 10 [    set pcolor green    set directions [-1 -1 180 -1]
    set velocity 1
set is-processing? FALSE    set failure FALSE    set output TRUE    set ID
    "BinPrH-spring"
set bifurcation FALSE ]
ask patch -7 10 [    set pcolor green    set directions [-1 -1 180 -1]
    set velocity 1
set is-processing? FALSE    set failure FALSE    set output TRUE    set ID
    "BinPrH-cover"
set bifurcation FALSE ]
end
;;;;;;;;;; corners ;;;;;;;;;;;
to setupCorners
ask patch 10 0 [    set pcolor brown    set directions [0 -1 -1 -1]    set
    velocity 1

```

```

set is-processing? FALSE      set failure FALSE      set bifurcation FALSE
  set ID "corner1" ]
ask patch 10 4 [      set pcolor brown      set directions [-1 -1 -1 270]
  set velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE
  set ID "corner2" ]
ask patch -10 4 [      set pcolor brown      set directions [-1 -1 180 -1]
  set velocity 1
set is-processing? FALSE      set failure FALSE      set bifurcation FALSE
  set ID "corner3" ]
end
;;;;;;;;; Labels ;;;;;;;;;;
to setupLabels
ask patch -11 0 [ set plabel "IN "      set plabel-color 102 ]
ask patch -14 1 [ set plabel "OUT"      set plabel-color 102 ]
ask patch -9 5 [ set plabel "QCWS"      set plabel-color 102 ]
ask patch 0 -1 [ set plabel "TeWS2"      set plabel-color 102 ]
ask patch -3 2 [ set plabel "TePaIWS"    set plabel-color 102 ]
ask patch -4 -1 [ set plabel "DWS"      set plabel-color 102 ]
ask patch -5 -9 [ set plabel "Stacking"  set plabel-color 102 ]
ask patch -5 -10 [ set plabel "magazine"  set plabel-color 102 ]
ask patch 0 3 [ set plabel "TeWS1"      set plabel-color 102 ]
ask patch 3 1 [ set plabel "IWS"      set plabel-color 102 ]
ask patch 7 1 [ set plabel "SWS"      set plabel-color 102 ]
ask patch 9 -3 [ set plabel "SWSOUT"    set plabel-color 102 ]
ask patch 5 5 [ set plabel "PWS"      set plabel-color 102 ]
ask patch 7 7 [ set plabel "CNa"      set plabel-color 102 ]
ask patch 3 7 [ set plabel "CNb"      set plabel-color 102 ]
ask patch 7 11 [ set plabel "RMTrcb"    set plabel-color 102 ]
ask patch 3 11 [ set plabel "RMTacb"    set plabel-color 102 ]
ask patch 0 7 [ set plabel "AWS"      set plabel-color 102 ]
end
;; Run Procedures ;;
to newTH-CO
let quantity read-from-string
user-input "How many products would you like?"
set quaCusOrd quantity
let cylBody runresult
[ user-one-of "What cylinder body color should the products be?"
["red" "cyan" "gray" ] ]
let piston runresult
[ user-one-of "What piston type should the products be?"
["aluminium" "black" ] ]

```

```

if cylBody = "red" and piston = "aluminium"
[ let quaRcbAp item 0 quaRcb
set quaRcb replace-item 0 quaRcb (quaRcbAp + quantity)
set totalRcbAp totalRcbAp + quantity ]
if cylBody = "red" and piston = "black"
[ let quaRcbBp item 1 quaRcb
set quaRcb replace-item 1 quaRcb (quaRcbBp + quantity)
set totalRcbBp totalRcbBp + quantity ]
if cylBody = "cyan" and piston = "aluminium"
[ let quaBcbAp item 0 quaBcb
set quaBcb replace-item 0 quaBcb (quaBcbAp + quantity)
set totalBcbAp totalBcbAp + quantity ]
if cylBody = "cyan" and piston = "black"
[ let quaBcbBp item 1 quaBcb
set quaBcb replace-item 1 quaBcb (quaBcbBp + quantity)
set totalBcbBp totalBcbBp + quantity ]
if cylBody = "gray" and piston = "aluminium"
[ let quaAcbAp item 0 quaAcb
set quaAcb replace-item 0 quaAcb (quaAcbAp + quantity)
set totalAcbAp totalAcbAp + quantity ]
if cylBody = "gray" and piston = "black"
[ let quaAcbBp item 1 quaAcb
set quaAcb replace-item 1 quaAcb (quaAcbBp + quantity)
set totalAcbBp totalAcbBp + quantity ]
ask patch -14 -2 [ set cusOrdNum cusOrdNum + 1 ]
set-default-shape TH-CustOrder "square"
create-TH-CustOrder 1 [ setxy -14 -2 set size 1 set heading 180
if cylBody = "red" and piston = "aluminium"
[ set shape "prh-cb-ap-sp-co" set color red set piston "aluminium"
" ]
if cylBody = "red" and piston = "black"
[ set shape "prh-cb-bp-sp-co" set color red set piston "black"
]
if cylBody = "cyan" and piston = "aluminium"
[ set shape "prh-cb-ap-sp-co" set color cyan set piston "
aluminium" ]
if cylBody = "cyan" and piston = "black"
[ set shape "prh-cb-bp-sp-co" set color cyan set piston "black"
]
if cylBody = "gray" and piston = "aluminium"
[ set shape "prh-cb-ap-sp-co" set color gray set piston "
aluminium" ]
if cylBody = "gray" and piston = "black"

```

```

[ set shape "prh-cb-bp-sp-co"      set color gray      set piston "black"
  ]
set birth-time ticks;
ask TH-CustOrder [ if any? TH-CustOrder-on patch-here [ jump 1
  ] ]
set thData [ -1 ]      set thData lput cusOrdNum thData      set thData lput
  quantity thData
set thData lput cylBody thData      set thData lput piston thData
set thData remove -1 thData      set thCoList lput cusOrdNum thCoList ]
user-message (word "Customer Order n. " cusOrdNum " for " quaCusOrd "
  products with "
cylBody " cylinder body and " piston " piston.")
end
;;;;;;;; to go ;;;;;;;;;;
to go
TH-workOrder[] TH-TrWS-pallet TH-TrWS-cylinderBody TH-TrWS-piston
SuH-WsOccupationTime SuH-WsProcessing SuH-countOutput SuH-statistics
tick
end
;;;;;;;;; Exemplo no texto – Listagem 2 – TH-workOrder[] ;;;;;;;;;;
to TH-workOrder [ PrH-pallet-type ] ;;Representa o início de uma ordem de
  produção.
set-default-shape PrH-pallet "square" ;;Define o formato do agente PrH-
  pallet.
ask patch -5 2 ;;Requisita ao agente OpH-TrWS a condição entre [].
[ ifelse count PrH-pallet-here = 0 ;;se não houver nenhum pallet neste
  agente, execute entre [].
[ set In-Free-BufferDWS TRUE ] ;;muda o valor da variável se o buffer da
  DWS não tem um PrH-cylinder body.
[ set In-Free-BufferDWS FALSE ] ] ;;muda o valor da variável se o buffer da
  DWS tem um PrH-cylinder body.
if In-Free-BufferDWS ;;se o buffer da DWS não tem um PrH-cylinder body,
  execute entre [].
[ create-PrH-pallet 1 ;;cria um PrH-pallet, conforme condição entre []
[ setxy -5 2 ;;no patch -5 2.
set size 1 ;;tamanho de 1 unidade
set heading 90 ;;direção da visão do agente, i.e., agente voltado para
  direita
set color Blue ;;cor do agente
set birth-time ticks; ;;considera o início do ciclo de vida do agente como
  sendo o tick em que foi criado.
set productionPlan ;;define o plano de produção inicialdo PrH que está
  entre []

```

```
[ "DWS" "TeWS" "IWS" "SWS" "PWS" "AWS" "QCWS" "OUT" ]
set processing FALSE ;;inicia o valor da variável ‘‘processing’’, para
monitorar se WS está processando o PrH
set QC-check "GOOD" ;;inicia o valor da variável ‘‘QC-check’’, para
monitorar defeitos no PrH.
set priority "new" ;;inicia o valor da variável ‘‘priority’’, para
monitorar a prioridade de execução do PrH
] ]

;;;;;;;;;;;;;;;;;;;;;;;;; Exemplo no texto – Listagem 2 – TH-
workOrder[] ;;;;;;;;;;;;;;;;;;;;;;;;;;
set-default-shape PrH-cylinderBody "cylinder"
ask patch -5 -8 [ ifelse count PrH-cylinderBody-here = 0
[ set IN-free-PrH-cylinderBody TRUE ] [ set IN-free-PrH-cylinderBody
FALSE ] ]
if IN-free-PrH-cylinderBody [ create-PrH-cylinderBody 1 [ setxy -5 -8
set size 0.9
set heading 0 set color one-of [red cyan gray ]
set productionPlan ["DWS" ] set processing FALSE ] ]
;;;;;;;;;;;;;;;;; Create PrH-piston ;;;;;;;;;;;;;;;;;
set-default-shape PrH-pistonAp "piston" set-default-shape PrH-pistonBp "
piston"
set-default-shape PrH-spring "lightning" set-default-shape PrH-cover "
target"
;;;;;;;;;;;;;;;;;;;;;;;;;
ask patch -1 10 [ ifelse count PrH-pistonAp-here = 0 [ set binApFree TRUE
] [ set binApFree FALSE ] ]
if binApFree [ create-PrH-pistonAp 1 [ setxy -1 10 set size 1 set
heading 180 set color yellow
set productionPlan ["AWS"] set processing FALSE ] ]
;;;;;;;;;;;;;;;;;;;;;;;;;
ask patch -3 10 [ ifelse count PrH-pistonBp-here = 0 [ set binBpFree TRUE
] [ set binBpFree FALSE ] ]
if binBpFree [ create-PrH-pistonBp 1 [ setxy -3 10 set size 1 set
heading 180 set color black
set productionPlan ["AWS"] set processing FALSE ] ]
;;;;;;;;;;;;;;;;;;;;;;;;;
ask patch -5 10 [ ifelse count PrH-spring-here = 0 [ set binSpFree TRUE ]
[ set binSpFree FALSE ] ]
if binSpFree [ create-PrH-spring 1 [ setxy -5 10 set size 1 set
heading 180 set color turquoise
set productionPlan ["AWS"] set processing FALSE ] ]
;;;;;;;;;;;;;;;;;;;;;;;;;
```



```

ask patch -7 10 [ ifelse count PrH-cover-here = 0 [ set binPrH-coverFree
  TRUE ] [ set binPrH-coverFree FALSE ] ]
if binPrH-coverFree [ create-PrH-cover 1 [ setxy -7 10 set size 1
  set heading 180 set color gray
set productionPlan ["AWS"] set processing FALSE ] ]
end
;;;;;;;;;; TrWS-cylinderBody movement ;;;;;;;;;;
to TH-TrWS-cylinderBody
ask PrH-cylinderBody [ ask patch-here [ set distance-to-jump
  velocity
set new_directions remove -1 directions ]
if not any? PrH-cylinderBody-on patch-ahead distance-to-jump and [ID] of
  patch-here != "DWS"
[ jump 0.05 ]
if [ID] of patch-here = "DWS" [ if color = red [set targetColorPrH-
  pallet red]
if color = cyan [set targetColorPrH-pallet cyan] if color = gray [set
  targetColorPrH-pallet gray] ] ]
end
;;;;;;;;;; TH-TrWS-piston movement ;;;;;;;;;;
to TH-TrWS-piston
ask PrH-pistonAp [ if not processing and [ID] of patch-here != "BinApExit"
[ ask patch-here [ set distance-to-jump velocity set
  new_directions remove -1 directions ]
if not any? PrH-pistonAp-on patch-ahead distance-to-jump
and [ID] of patch-here != "BinApExit" and not [is-processing?] of patch-
  ahead distance-to-jump
[ jump distance-to-jump ] ] if any? PrH-pallet-on patch-here [ die
  ] ]
ask PrH-pistonBp [ if not processing and [ID] of patch-here != "BinBpExit
"
[ ask patch-here [ set distance-to-jump velocity set
  new_directions remove -1 directions ]
if not any? PrH-pistonBp-on patch-ahead distance-to-jump and ID != "
  BinBpExit"
and not [is-processing?] of patch-ahead distance-to-jump [ jump distance-
  to-jump ] ]
if any? PrH-pallet-on patch-here [ die ] ]
ask PrH-spring [ if not processing and [ID] of patch-here != "BinSpExit"
[ ask patch-here [ set distance-to-jump velocity set
  new_directions remove -1 directions ]
if not any? PrH-spring-on patch-ahead distance-to-jump and ID != "
  BinSpExit"

```

```

and not [is-processing?] of patch-ahead distance-to-jump [ jump distance-
to-jump ] ]
if any? PrH-pallet-on patch-here [ die ] ]
ask PrH-cover [ if not processing and [ID] of patch-here != "BinPrH-
coverExit"
[ ask patch-here [ set distance-to-jump velocity set
new_directions remove -1 directions ]
if not any? PrH-cover-on patch-ahead distance-to-jump and ID != "BinPrH-
coverExit"
and not [is-processing?] of patch-ahead distance-to-jump [ jump distance-
to-jump ] ]
if any? PrH-pallet-on patch-here [ die ] ]
end
;;;;;;;;; Exemplo no texto – Listagem 3: TH-TrWS-pallet ;;;;;;;;;;
to TH-TrWS-pallet ;;Controla o movimento do pallet
ask PrH-pallet ;;requisita ao PrH-pallet entre []
[ ask patch-here ;;requisita ao patch atual entre []
[ set distance-to-jump velocity ;;muda a velocidade do PrH
set new_directions remove -1 directions ;;define qual a direção do PrH, no
atual patch.
]
;;;;;;;;;
set heading first new_directions
;;;;;;;;; next ==> DWS ;;;;;;;;;;
if not processing and first productionPlan = "DWS" and [ID] of patch-
ahead distance-to-jump != "DWS"
and not any? PrH-pallet-on patch-ahead distance-to-jump [ jump distance-
to-jump ]
;;;;;;;;; DWS ;;;;;;;;;;
if not processing and first productionPlan = "DWS" and [ID] of patch-
ahead distance-to-jump = "DWS"
and not any? PrH-pallet-on patch-ahead distance-to-jump [ jump distance-to-
-jump
set productionPlan remove-item 0 productionPlan set productionPlan fput
"PaDWS" productionPlan
set heading item 1 directions set processing TRUE set is-processing?
TRUE
set time-of-processing processing-time set TimeDWS TimeDWS + processing-
time ]
;;;;;;;;; PalletWS ;;;;;;;;;;
if not processing and first productionPlan = "PaDWS" and [ID] of patch-
here = "DWS"

```

```

and any? PrH-cylinderBody-on patch-here [ set productionPlan remove-item 0
  productionPlan
set color targetColorPrH-pallet set shape "prh-cylinder"
ask PrH-cylinderBody-on patch-here [ die ] ]
;;;;;DWS ==> TeWS ;;;;;;;;;;;;;;
if not processing and first productionPlan = "TeWS"
and [ID] of patch-ahead distance-to-jump != "InTeWS"
and not [failure] of patch-ahead distance-to-jump
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ]
;;;;;TeWS ==> InTeWS ;;;;;;;;;;;;;;
if not processing and first productionPlan = "TeWS"
and [ID] of patch-ahead distance-to-jump = "InTeWS"
and not [failure] of patch-ahead distance-to-jump
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ]
;;;;; Exemplo no texto – Listagem 4 ;;;;
;;;;;;;;;;;;; Here InTeWS ;;;;;;;;;;
if not processing ;condição para o agente requisitar o OpH-TeWS
and first productionPlan = "TeWS" ;condição de verificação do plano de
  produção
and [ID] of patch-here = "InTeWS" ;condição se o patch atual é a bifurcação
  que representa a entrada da TeWS, execute[]
[ if not [failure-TeWS1] of patch-at 2 2 ;condição se não houver falha na
  TeWS1
and not use-TeWS2-Manual ;e se a TeWS2 estiver liberada para uso em seleção
  manual
[ set heading item 0 directions ;põe o agente PrH-pallet na 1a direção da
  lista , neste caso vertical de cima para baixo
if not any? PrH-pallet-on patch-ahead distance-to-jump ;verifica se tem
  algum PrH-pallet no próximo patch
[ jump distance-to-jump ;desloca o agente da distância definida
set productionPlan remove "TeWS" productionPlan ;modifica o plano de
  produção retirando o plano TeWS
set productionPlan fput "TeWS1" productionPlan ;inclui o plano TeWS1
]
]
;;;;;;;;;;;;;
if [failure-TeWS1] of patch-at 2 2 [ set heading item 2 directions
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump
set productionPlan remove "TeWS" productionPlan
set productionPlan fput "TeWS2" productionPlan ] ]

```



```

set TimeTeWS2 TimeTeWS2 + processing-time    ]
;;;;TeWS ==> IWS ;;;;;;;;;;;;;;
if not processing and first productionPlan = "IWS"
and [ID] of patch-ahead distance-to-jump != "IWS"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ]
;;;;; next ==> IWS ;;;;;;;;;;;;;;
if not processing and first productionPlan = "IWS"
and [ID] of patch-ahead distance-to-jump = "IWS"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ]
set productionPlan remove-item 0 productionPlan
set processing TRUE      set is-processing? TRUE
set time-of-processing processing-time
set TimeIWS TimeIWS + processing-time    ]
;;;;;IWS ==> SWS ;;;;;;;;;;;;;;
if not processing and first productionPlan = "SWS"
and [ID] of patch-ahead distance-to-jump != "SWS"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ]
;;;;; next ==> SWS ;;;;;;;;;;;;;;
if not processing and [ID] of patch-ahead distance-to-jump = "SWS"
and not any? PrH-pallet-on patch-ahead distance-to-jump
and ( priority = "new" or priority = "toAWS" )
[ if ( item 0 quaBcb > 0 or item 1 quaBcb > 0 ) and color = cyan
[ jump distance-to-jump ]
set productionPlan remove-item 0 productionPlan
set processing TRUE      set is-processing? TRUE
set time-of-processing processing-time
set TimeSWS TimeSWS + processing-time    ]
if ( item 0 quaBcb <= 0 and item 1 quaBcb <= 0 ) and color = cyan
[ jump distance-to-jump ]
set productionPlan fput "SWSOUT" productionPlan
set productionPlan remove "PWS" productionPlan
set productionPlan remove "AWS" productionPlan
set productionPlan remove "QCWS" productionPlan
set productionPlan remove "OUT" productionPlan
set processing TRUE      set is-processing? TRUE
set time-of-processing processing-time
set TimeSWS TimeSWS + processing-time
set heading item 2 directions    ]
;;;;;;;;;;;;; Exemplo no texto - Listagem 6 ;;;;;;;;;;;;;;

```



```

and [ID] of patch-ahead distance-to-jump = "SWS"
and not any? PrH-pallet-on patch-ahead distance-to-jump
and priority = "away"
[ jump distance-to-jump
set productionPlan fput "SWSOUT" productionPlan
set productionPlan remove "SWS" productionPlan
set processing TRUE      set is-processing? TRUE
set time-of-processing processing-time
set TimeSWS TimeSWS + processing-time ]
;;;; SWS => SWSOUT ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
if not processing      and first productionPlan = "SWSOUT"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ set heading item 2 directions      jump distance-to-jump      ]
;;;;;SWS OUT ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
if not processing      and [ID] of patch-here = "SWSOUT"      [ die      ]
;;;;;SWS => InPWS ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
if not processing      and first productionPlan = "PWS"
and [ID] of patch-ahead distance-to-jump != "InPWS"
and [ID] of patch-ahead distance-to-jump != "InCNC"
and not any? PrH-pallet-on patch-ahead distance-to-jump
and priority != "toAws" [ jump distance-to-jump ]
;;;;;;;;;;;;;;;;; next => InPWS ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
if not processing      and first productionPlan = "PWS"
and [ID] of patch-ahead distance-to-jump = "InPWS"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ if item 0 quaAcb > 0 and color = gray
[ jump distance-to-jump
set productionPlan fput "InCNC" productionPlan
set productionPlan remove "PWS" productionPlan
set processing TRUE      set is-processing? TRUE
set heading item 0 directions      ]
if item 1 quaAcb > 0 and color = gray
[ jump distance-to-jump
set productionPlan fput "InCNC" productionPlan
set productionPlan remove "PWS" productionPlan
set processing TRUE      set is-processing? TRUE
set heading item 0 directions      ]
if item 0 quaBcb > 0 and color = cyan
[ jump distance-to-jump
set productionPlan fput "InCNC" productionPlan
set productionPlan remove "PWS" productionPlan
set processing TRUE      set is-processing? TRUE
set heading item 0 directions      ]

```

```

if item 1 quaBcb > 0 and color = cyan
[ jump distance-to-jump
set productionPlan fput "InCNC" productionPlan
set productionPlan remove "PWS" productionPlan
set processing TRUE      set is-processing? TRUE
set heading item 0 directions ]
if item 0 quaRcb > 0 and color = red
[ jump distance-to-jump
set productionPlan fput "InCNC" productionPlan
set productionPlan remove "PWS" productionPlan
set processing TRUE      set is-processing? TRUE
set heading item 0 directions ]
if item 1 quaRcb > 0 and color = red
[ jump distance-to-jump
set productionPlan fput "InCNC" productionPlan
set productionPlan remove "PWS" productionPlan
set processing TRUE      set is-processing? TRUE
set heading item 0 directions ]
if item 0 quaAcb <= 0 and item 1 quaAcb <= 0 and color = gray
[ set priority "away"
set productionPlan fput "SWS" productionPlan
set productionPlan remove "PWS" productionPlan
set productionPlan remove "AWS" productionPlan
set productionPlan remove "QCWS" productionPlan
set productionPlan remove "OUT" productionPlan
jump distance-to-jump
set heading item 3 directions ]
if item 0 quaBcb <= 0 and item 1 quaBcb <= 0 and color = cyan
[ set priority "away"
set productionPlan fput "SWS" productionPlan
set productionPlan remove "PWS" productionPlan
set productionPlan remove "AWS" productionPlan
set productionPlan remove "QCWS" productionPlan
set productionPlan remove "OUT" productionPlan
jump distance-to-jump
set heading item 3 directions ]
if item 0 quaRcb <= 0 and item 1 quaRcb <= 0 and color = red
[ set priority "away"
set productionPlan fput "SWS" productionPlan
set productionPlan remove "PWS" productionPlan
set productionPlan remove "AWS" productionPlan
set productionPlan remove "QCWS" productionPlan
set productionPlan remove "OUT" productionPlan

```



```

jump distance-to-jump
set heading item 3 directions ] ]
; InPWS => InCNC
; ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
if not processing and first productionPlan = "InCNC"
and [ID] of patch-here != "InCNC"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ]
; ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
if not processing and first productionPlan = "InCNC"
and [ID] of patch-here = "InCNC"
[ if Color = red
[ set heading item 0 directions
set productionPlan remove "InCNC" productionPlan
set productionPlan fput "InAWS" productionPlan
set productionPlan fput "CNC B" productionPlan
set productionPlan fput "RMT rcb" productionPlan
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ]
if Color = gray
[ set heading item 3 directions
set productionPlan remove "InCNC" productionPlan
set productionPlan fput "InAWS" productionPlan
set productionPlan fput "RMT acb" productionPlan
set productionPlan fput "CNC A" productionPlan
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ]
if Color = cyan
[ set heading item 3 directions
ifelse not any? PrH-pallet-on patch-ahead 6
[ set productionPlan remove "InCNC" productionPlan
set productionPlan fput "InAWS" productionPlan
set productionPlan fput "CNC B" productionPlan
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ]
[ set productionPlan remove "InCNC" productionPlan
set productionPlan fput "InAWS" productionPlan
set productionPlan fput "CNC A" productionPlan
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ] ] ]
; :: next ==> WS; ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
if not processing and first productionPlan = "CNC A"
and [ID] of patch-ahead distance-to-jump = "CNC A"

```

```

and not any? PrH-pallet-on patch-ahead distance-to-jump
and [ID] of patch-here != "InCNC"
[ jump distance-to-jump
set productionPlan remove-item 0 productionPlan
set processing TRUE      set is-processing? TRUE
set time-of-processing processing-time
set TimeCNC_A TimeCNC_A + processing-time ]
;;;;;;;; next ==> CNCA ;;;;;;;;;
if not processing and first productionPlan = "CNC A"
and [ID] of patch-here = "InCNC"
[ if Color = gray
[ set heading item 3 directions
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ]
if Color = cyan
[ set heading item 3 directions
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ]
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ] ]
;;;;;;;; next ==> InCNCB ;;;;;;;;;
if not processing and [ID] of patch-here = "InCNCB"
[ if ( Color = gray )
[ set heading item 3 directions
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ]
if ( Color = cyan )
[ set heading item 3 directions
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ]
if ( Color = red )
[ set heading item 2 directions
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ] ] ]
;;;;;;;; ==> CNCB ;;;;;;;;;
if not processing and first productionPlan = "CNC B"
and [ID] of patch-ahead distance-to-jump != "CNC B"
and [ID] of patch-ahead distance-to-jump != "InCNC B"
and [ID] of patch-here != "InCNC"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ]
if not processing and first productionPlan = "CNC B"
and [ID] of patch-here = "InCNC"

```

```

and ( color = cyan or color = gray )
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ set heading item 3 directions
jump distance-to-jump ]
;;;;; next ==> CNCB ;;;;;;;;;;;;;;
if not processing and first productionPlan = "CNC B"
and [ID] of patch-ahead distance-to-jump = "CNC B"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump
set productionPlan remove "CNC B" productionPlan
set processing TRUE
set is-processing? TRUE
set time-of-processing processing-time
set TimeCNC_B TimeCNC_B + processing-time ]
;;;;;;;;;
if not processing and first productionPlan = "InAWS"
and [ID] of patch-ahead distance-to-jump = "InRMTacb"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ]
;;;;;;;;;
if [ID] of patch-here = "InRMTacb"
[ if color = "gray" [ set heading item 0 directions
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ] ]
if color = cyan [ if not processing and first productionPlan = "InAWS"
[ set heading item 2 directions
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ] ]
if not processing and first productionPlan = "CNC B"
[ set heading item 3 directions
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ] ] ]
;;;;; next ==> WS ;;;;;;;;;;;;;;
if not processing and first productionPlan = "RMT acb"
and [ID] of patch-ahead distance-to-jump = "RMT acb"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump
set productionPlan remove "RMT acb" productionPlan
set processing TRUE set is-processing? TRUE
set time-of-processing processing-time
set TimeRMT_acb TimeRMT_acb + processing-time ]
;;;;; next ==> RMT rcb
;;;;;;;;;

```

```

if not processing and first productionPlan = "RMT acb"
and [ID] of patch-ahead distance-to-jump != "RMT acb"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ]
;;;;; next ==> RMT rcb

; ; ; ; ;
if not processing and first productionPlan = "RMT rcb"
and [ID] of patch-ahead distance-to-jump != "RMT rcb"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ]
;;;;; next ==> WS; ; ; ; ;
if not processing and first productionPlan = "RMT rcb"
and [ID] of patch-ahead distance-to-jump = "RMT rcb"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump
set productionPlan remove "RMT rcb" productionPlan
set processing TRUE set is-processing? TRUE
set time-of-processing processing-time
set TimeRMT_rcb TimeRMT_rcb + processing-time ]
; ; ; ; ;
if not processing and first productionPlan = "InAWS"
and [ID] of patch-ahead distance-to-jump != "InAWS"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ]
;;;TeWS ==> IWS ; ; ; ; ;
if not processing and first productionPlan = "InAWS"
and [ID] of patch-ahead distance-to-jump = "InAWS"
and not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump
set heading item 0 directions
ifelse not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump
set productionPlan remove "InAWS" productionPlan ]
[ set heading item 3 directions
set productionPlan fput "PriorAWS" productionPlan
set productionPlan remove "InAWS" productionPlan
set priority "toAws"
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ] ]
;;;TeWS ==> IWS ; ; ; ; ;
if not processing and first productionPlan = "PriorAWS"
and [ID] of patch-here != "InAWS"
and not any? PrH-pallet-on patch-ahead distance-to-jump

```

```

[ jump distance-to-jump ]
;;;; PriorAws ;;;;;;;;;;;;;;
if not processing and first productionPlan = "PriorAWS"
and [ID] of patch-here = "InAWS"
[ set heading item 0 directions
ifelse not any? PrH-pallet-on patch-ahead distance-to-jump
[ set productionPlan remove "PriorAWS" productionPlan jump distance-to-
jump]
[ set heading item 3 directions set priority "toAws"
if not any? PrH-pallet-on patch-ahead distance-to-jump
[ jump distance-to-jump ] ] ]
;;;;; next ==> SWS ;;;;;;;;;;;;;;
if not processing and first productionPlan = "AWS"
and [ID] of patch-ahead distance-to-jump = "AWS"
[ if not any? PrH-pallet-on patch-ahead distance-to-jump
[ set productionPlan remove "AWS" productionPlan
jump distance-to-jump
if ( item 0 quaAcb >= item 1 quaAcb )
and item 0 quaAcb > 0 and color = gray
[ set productionPlan fput "BinApExit" productionPlan ]
if ( item 0 quaAcb < item 1 quaAcb )
and item 1 quaAcb > 0 and color = gray
[ set productionPlan fput "BinBpExit" productionPlan ]
if ( item 0 quaBcb >= item 1 quaBcb ) and item 0 quaBcb > 0
and color = cyan
[ set productionPlan fput "BinApExit" productionPlan ]
if ( item 0 quaBcb < item 1 quaBcb ) and item 1 quaBcb > 0
and color = cyan
[ set productionPlan fput "BinBpExit" productionPlan ]
if ( item 0 quaRcb >= item 1 quaRcb ) and item 0 quaRcb > 0
and color = red
[ set productionPlan fput "BinApExit" productionPlan ]
if ( item 0 quaRcb < item 1 quaRcb ) and item 1 quaRcb > 0
and color = red
[ set productionPlan fput "BinBpExit" productionPlan ]
set heading item 3 directions
set processing TRUE set is-processing? TRUE
set time-of-processing processing-time
set TimeAWS TimeAWS + processing-time
if item 0 quaAcb <= 0 and item 1 quaAcb <= 0 and color = gray
[ set priority "away"
set productionPlan fput "SWS" productionPlan
set productionPlan remove "AWS" productionPlan

```



```

[ if ID = "TeWS1"      [ set processing-time TeWS1-processing-time      ]
if ID = "TeWS2" [ set processing-time TeWS2-processing-time      ]
if ID = "CNC A"  [ ifelse ( item 0 quaAcb + item 1 quaAcb ) < 50
[ set processing-time 40      ]      [ set processing-time 20      ]      ]
if ID = "CNC B"
[ ifelse ( item 0 quaRcb + item 1 quaRcb ) < 50
[ set processing-time 40      ]      [ set processing-time 20      ] ] ]
;;;;;;;;;;;;;;;;;;;;;;;;;
ask PrH-pallet
[ if processing      [ set time-of-processing time-of-processing - 1
ask patch-left-and-ahead 90 1
[ set plabel-color Red      ]
if [ID] of patch-here != "TeWS1"
[ ask patch-right-and-ahead 90 1      [ set plabel-color Red      ]
]
ifelse [ID] of patch-here = "CNC A" or      [ID] of patch-here = "CNC B" or
[ID] of patch-here = "RMT rcb" or      [ID] of patch-here = "RMT acb"
[ ask patch 5 5      [ set plabel-color Red      ]      ]
[ ask patch 5 5      [ set plabel-color Black      ]      ]
ask patch-ahead 1      [ set plabel-color Red      ]
if [ID] of patch-here = "CNC A"      [ ask patch 5 5
[ set plabel-color Red      ]      ]
if [ID] of patch-here = "DWS"      [ ask patch -4 -1
[ set plabel-color Red      ]      ] ] ]
ask PrH-pallet [ if processing      [ if time-of-processing <= 0
[ ask patch-here      [ set is-processing? FALSE      ]
ask patch-left-and-ahead 90 1
[ set plabel-color 102      ]
set processing FALSE
ask patch-right-and-ahead 90 1      [ set plabel-color 102      ]
ask patch-ahead 1      [ set plabel-color 102      ]
ask patch -4 -1      [ set plabel-color 102      ]
if [ID] of patch-here = "IWS"
[ ifelse random-float 1 < 0.1
[ set QC-check "BAD"      ]      [ set QC-check "GOOD"      ]
]
;;;;;;;;;;;;;;;;;;;;;;;;;Exemplo no texto - Listagem 5;;;;;;;;;;;;;;;;;;;;;;;;;
if [ID] of patch-here = "QCWS" ;;verificação de controle de qualidade
[ ifelse random-float 1 < 0.1 ;;definição de função aleatória
[ set QC-check "BAD" ;;se a condição após ifelse for verdadeira altera a
variável QC-check para BAD
]
]

```

```

[ set QC-check "GOOD" ;;se a condição após ifelse for falsa altera a
  variável QC-check para GOOD
]
]
]
]
]
]
ifelse failure-TeWS1 [ set use-TeWS2-Automatic TRUE
ask patch 0 2 [ set failure TRUE ] ]
[ set use-TeWS2-Automatic FALSE
ask patch 0 2 [ set failure FALSE ] ]
end
;;;;;;;;;;;;; Exemplo no texto – Listagem 7 ;;;;;;;;;;;;;;
;;monitora a quantidade de produto prh-bcb-ap-sp-co produzida no sistema
to SuH-countOutput
ask PrH-pallet
[ if [ID] of patch-here = "OUT" [ set death-time ticks
ifelse QC-check = "GOOD"
[ set FinalProduct-outputTotal FinalProduct-outputTotal + 1
if shape = "prh-cb-ap-sp-co" and color = cyan
[ set FinalProduct-outputPrH_bcb_ap FinalProduct-outputPrH_bcb_ap + 1
let quaBcbAp item 0 quaBcb - 1
set quaBcb replace-item 0 quaBcb quaBcbAp ]
;
;;;;;;;;;;;;;

if shape = "prh-cb-ap-sp-co" and color = red
[ set FinalProduct-outputPrH_rcb_ap FinalProduct-outputPrH_rcb_ap + 1
let quaRcbAp item 0 quaRcb - 1
set quaRcb replace-item 0 quaRcb quaRcbAp ]
if shape = "prh-cb-ap-sp-co" and color = gray
[ set FinalProduct-outputPrH_acb_ap FinalProduct-outputPrH_acb_ap + 1
let quaAcbAp item 0 quaAcb - 1
set quaAcb replace-item 0 quaAcb quaAcbAp ]
if shape = "prh-cb-bp-sp-co" and color = cyan
[ set FinalProduct-outputPrH_bcb_bp FinalProduct-outputPrH_bcb_bp + 1
let quaBcbBp item 1 quaBcb - 1
set quaBcb replace-item 1 quaBcb quaBcbBp ]
if shape = "prh-cb-bp-sp-co" and color = red
[ set FinalProduct-outputPrH_rcb_bp FinalProduct-outputPrH_rcb_bp + 1
let quaRcbBp item 1 quaRcb - 1
set quaRcb replace-item 1 quaRcb quaRcbBp ]
if shape = "prh-cb-bp-sp-co" and color = gray

```

```

[ set FinalProduct-outputPrH_acb_bp FinalProduct-outputPrH_acb_bp + 1
let quaAcbBp item 1 quaAcb - 1
set quaAcb replace-item 1 quaAcb quaAcbBp ]
if shape = "prh-cylinder" and color = cyan
[ set FinalProduct-outputPrH_bcb FinalProduct-outputPrH_bcb + 1 ]
if shape = "prh-cylinder" and color = red
[ set FinalProduct-outputPrH_rcb FinalProduct-outputPrH_rcb + 1 ]
if shape = "prh-cylinder" and color = gray
[ set FinalProduct-outputPrH_acb FinalProduct-outputPrH_acb + 1 ]
;;;;;;;;;;;;;SuH-Agenda;;;;;;;;;;;;;
ask TH-CustOrder
[ if shape = "prh-cb-ap-sp-co" and item 2 thData = "gray"
and item 3 ThData = "aluminium"
and totalAcbAp = FinalProduct-outputPrH_acb_ap
[ set thCoList remove cusOrdNum thCoList
user-message (word "Customer Order n. "
item 0 ThData " for " item 1 ThData " products with "
item 2 ThData " cylinder body and " item 3 ThData " piston is completed"
)
die ]
if shape = "prh-cb-bp-sp-co" and item 2 thData = "gray"
and item 3 ThData = "black"
and totalAcbBp = FinalProduct-outputPrH_acb_bp
[ set thCoList remove cusOrdNum thCoList
user-message (word "Customer Order n. "
item 0 ThData " for " item 1 ThData " products with "
item 2 ThData " cylinder body and " item 3 ThData " piston is completed"
)
die ]
if shape = "prh-cb-ap-sp-co" and item 2 thData = "cyan"
and item 3 ThData = "aluminium"
and totalBcbAp = FinalProduct-outputPrH_bcb_ap
[ set thCoList remove cusOrdNum thCoList
user-message (word "Customer Order n. "
item 0 ThData " for " item 1 ThData " products with "
item 2 ThData " cylinder body and " item 3 ThData " piston is completed"
)
die ]
if shape = "prh-cb-bp-sp-co" and item 2 thData = "cyan"
and item 3 ThData = "black"
and totalBcbBp = FinalProduct-outputPrH_bcb_bp
[ set thCoList remove cusOrdNum thCoList
user-message (word "Customer Order n. "

```

```

item 0 ThData " for " item 1 ThData " products with "
item 2 ThData " cylinder body and " item 3 ThData " piston is completed"
)
die ]
if shape = "prh-cb-ap-sp-co" and item 2 thData = "red"
and item 3 ThData = "aluminium"
and totalRcbAp = FinalProduct-outputPrH_rcb_ap
[ set thCoList remove cusOrdNum thCoList
user-message (word "Customer Order n. "
item 0 ThData " for " item 1 ThData " products with "
item 2 ThData " cylinder body and " item 3 ThData " piston is completed"
)
die ]
if shape = "prh-cb-bp-sp-co" and item 2 thData = "red"
and item 3 ThData = "black"
and totalRcbBp = FinalProduct-outputPrH_rcb_bp
[ set thCoList remove cusOrdNum thCoList
user-message (word "Customer Order n. "
item 0 ThData " for " item 1 ThData " products with "
item 2 ThData " cylinder body and " item 3 ThData " piston is completed"
)
die ] ] ]
;;;;;;;;;;;;; Exemplo no texto – Listagem 7 ;;;;;;;;;;;;;;
;;monitora MLT
[ set FinalProduct-output-bad FinalProduct-output-bad + 1 ]
set mlt-aux (mlt-aux + (death-time - birth-time))
set avg-mlt (mlt-aux / (FinalProduct-outputTotal + FinalProduct-output-bad)
)
die ] ]
end
;;;;;;;;;;;;;
to SuH-WsOccupationTime
ask patches
[ ifelse any? PrH-pallet-here
[ set is-processing? TRUE ] [ set is-processing? FALSE ] ]
end
;;;;;;;;;;;;; Exemplo no texto – Listagem 7 ;;;;;;;;;;;;;;
;;monitora WiP
to SuH-statistics
set wip count PrH-pallet-on patches + count PrH-cylinderBody-on patches
set-current-plot-pen "DWS" plot TimeDWS
;;;;;;;;;;;;; Exemplo no texto – Listagem 8 ;;;;;;;;;;;;;;
;;;;;;;;;monitora tempo de processamento

```

```
set-current-plot-pen "TeWS1" plot TimeTeWS1
set-current-plot-pen "TeWS2" plot TimeTeWS2
set-current-plot-pen "IWS" plot TimeIWS
set-current-plot-pen "SWS" plot TimeSWS
set-current-plot-pen "RMT rcb" plot TimeRMT_rcb
set-current-plot-pen "RMT acb" plot TimeRMT_acb
set-current-plot-pen "CNC A" plot TimeCNC_A
set-current-plot-pen "CNC B" plot TimeCNC_B
set-current-plot-pen "AWS" plot TimeAWS
set-current-plot-pen "QCWS" plot TimeQCWS
set-current-plot-pen "TePalWS" plot TimeTePalWS
end
```

Apêndice B – Publicações Decorrentes do Trabalho

Para o desenvolvimento deste trabalho, uma fundamental e importante tarefa foi a produção científica que, entre outros, envolveu elaboração de artigos para revistas e eventos científicos. Desse modo, foi possível a divulgação e interação com pares acadêmicos/pesquisadores dos resultados parciais obtidos e a correção de rumos. Durante seu doutorado, o autor participou dos seguintes eventos:

- (i) IEEE 12th *International Conference on Industrial Informatics* (INDIN). Ano: 2014.
- (ii) 15th IFAC/IEEE/IFIP/IFORS *Symposium on Information Control Problems in Manufacturing*. Ano: 2015.
- (iii) Congresso Brasileiro de Automática (CBA). Ano: 2012, 2014.
- (iv) *Doctoral Conference on Computing, Electrical and Industrial Systems* (DoCEIS). Ano: 2011, 2014, 2015.
- (v) *International Congress of Mechanical Engineering*, COBEM. Ano: 2011, 2013 e 2015.
- (vi) VII Congresso Nacional de Engenharia Mecânica. CONEM. Ano: 2012.
- (vii) Simpósio Brasileiro de Automação Inteligente (SBAI) Ano: 2011, 2015

A participação em eventos científicos possibilitou ampliar gradativamente o desenvolvimento desta proposta. Por exemplo, no INDIN o autor realizou contatos com o líder de pesquisa que desenvolveu a IOPT; no INCOM 2015 observou-se que os modelos IOPT poderiam ser implementados usando o *software* Netlogo. No DoCEIS 2015, o autor conheceu pesquisadores que contribuíram direta ou indiretamente para o desenvolvimento do exemplo de aplicação. Outros eventos (por exemplo, COBEM 2013, 2015, CBA 2012, 2014, SBAI 2011, 2013, 2015, COBEF 2011) também trouxeram contribuições importantes e parcerias para desenvolvimento de projetos de pesquisa em comum. É possível observar que alguns dos trabalhos apresentados foram selecionados para publicação em periódicos ou capítulos livros (para maiores detalhes vide o currículo lattes do autor no *link* <http://lattes.cnpq.br/3307568955007464>). Alguns desses trabalhos, bem como algumas apresentações (inclusive a apresentação de defesa desta tese) foram disponibilizados no portal acadêmico *Research Gate*: http://www.researchgate.net/profile/Robson_Silva7.

Para entendimento da contribuição/desenvolvimento, lista-se a seguir a produção científica desenvolvida durante o doutoramento, comentando a relação de contribuição de cada uma

com a solução proposta neste trabalho e descrevendo os aspectos mais relevantes. Os artigos decorrentes da pesquisa publicados em periódicos ou capítulos de livro são:

- (i) da Silva, R. M.; Junqueira, Fabrício ; Santos Filho, Diolino J. ; Miyagi, P. E. **Control architecture and design method of reconfigurable manufacturing systems**. *Control Engineering Practice*, v. 49, p. 87-100, 2016. DOI: 10.1016/j.conengprac.2016.01.009. Disponível em <http://www.sciencedirect.com/science/article/pii/S0967066116300090>.
 - (a) O objetivo da revista internacional *Control Engineering Practice* é publicar artigos que demonstram a aplicação prática da teoria de controle e suas ferramentas em todas as possíveis áreas da automação industrial. O artigo publicado apresenta uma síntese geral deste trabalho e sua aceitação demonstra o papel relevante da proposta para a implementação do controle de Sistemas Reconfiguráveis de Manufatura;
- (ii) da Silva, R. M.; Watanabe, E. H. ; Blos, M. F.; Junqueira, F. ; Santos Filho, D. J. ; Miyagi, P. E. **Modeling of Mechanisms for Reconfigurable and Distributed Manufacturing Control System**. *IFIP Advances in Information and Communication Technology*, v. 450, p. 93-100, 2015.
 - (a) Esta publicação é resultante do trabalho apresentado no evento científico DoCEIS 2015. O artigo demonstra o desenvolvimento dos mecanismos de diagnóstico de falhas apresentado na Sub-seção 4.2.3;
- (iii) da Silva, Robson Marinho; Santos Filho, Diolino J. ; Miyagi, P. E. . **From Conception to Implementation of Reconfigurable and Distributed Manufacturing Control System**. *Lecture Notes in Computer Science*, v. 9266, p. 35-46, 2015.
 - (a) Esta publicação é resultante do trabalho apresentado para o evento científico HoloMAS 2015 que objetiva a interação de pesquisadores com trabalhos de pesquisa relevantes na área de HMAS e SOA para gerenciar, modelar e suportar o desenvolvimento de sistemas complexos industriais. O foco deste artigo é demonstrar como a abordagem usa ferramentas e conceitos levantados no Capítulo 2, principalmente àqueles relacionados a HMAS e SOA, e a partir da abordagem como desenvolver controle de SRMs, conforme apresentado no método proposto no Capítulo 4. A seleção do artigo para publicação na revista internacional *Lecture Notes in Computer Science* demonstra que o método proposto tem relevância dentro do objetivo do evento HoloMAS;
- (iv) Blos, Mauricio F. ; da Silva, Robson M. ; Miyagi, Paulo E. . **Application of an Agent-based Supply Chain to Mitigate Supply Chain Disruptions**. *IFAC-PapersOnLine*,

v. 48, p. 640-645, 2015.

- (a) Esta publicação é resultante do trabalho apresentado no evento científico INCOM 2015. O objetivo do artigo foi explorar como aspectos desta proposta podem ser aplicados em outros estudos de caso de sistemas de manufatura. Especificamente, este artigo aplicou os conceitos relacionados a MAS e RdP para identificar possíveis riscos e mitigar interrupções na cadeia de suprimento de um sistema de manufatura;
- (v) da Silva, Robson Marinho; Benítez-Pina, Israel F. ; Blos, Mauricio F. ; Filho, Diolino J. Santos ; Miyagi, Paulo E. . **Modeling of reconfigurable distributed manufacturing control systems**. IFAC-PapersOnLine, v. 48, p. 1284-1289, 2015.
 - (a) Esta publicação é resultante do trabalho apresentado no evento científico INCOM 2015. O foco do artigo está em demonstrar aspectos mais relevantes do Capítulo 4 considerando conceitos de sistemas dispersos;
- (vi) Silva, R.M.; Blos, M.F.; Junqueira, F.; Santos Filho, D.J.; Miyagi, P.E.; **A service-oriented and holonic control architecture for reconfiguration of dispersed manufacturing system** . IFIP Advances in Information and Communication Technology, 2014.
 - (a) Esta publicação é resultante do trabalho apresentado no evento científico DoCEIS 2014. Este artigo demonstra como os conceitos relacionados a SOA podem ser combinados com HMAS para aplicação em SRM;
- (vii) Silva, R.M.; Arakaki, J.; Junqueira, F.; Santos Filho, D.J.; Miyagi, P.E. **Modeling of active holonic control systems for intelligent buildings** . Automation in Construction (Print), v. 25, p. 20-33, 2012.
 - (a) Esta publicação é resultante do projeto de pesquisa realizado para ingressar no doutorado. Assim, abordou conceitos básicos propostos aqui, os quais posteriormente serviram de base para um aprofundamento e desenvolvimento desta proposta. A aplicação considerado foi em edifícios inteligentes, uma importante classe de sistema de produtivo, assim como são os sistemas de manufatura;
- (viii) Silva, R.M.; Arakaki, J.; Junqueira, F.; Santos Filho, D.J.; Miyagi, P.E. **A procedure for modeling of Holonic Control Systems for Intelligent** . Advanced Materials Research, v. 383-390, p. 2318-2326, 2012.

- (a) Esta publicação também é resultante do projeto de pesquisa realizado para ingressar no doutorado e foi selecionada pelo editorial do trabalho submetido ao evento ICEEAC 2015;
- (ix) Silva, R.M.; Miyagi, P.E.; Santos Filho, D.J. **Design of Active Holonic Fault-Tolerant Control Systems**. IFIP Advances in Information and Communication Technology, v. 349, p. 367-374, 2011.
- (a) Esta publicação é resultante do artigo enviado para o DoCEIS'2011. Aborda o desenvolvimento de técnicas de tomada de decisão para desenvolver um sistema de controle tolerante a falhas;
- (x) da Silva, Robson Marinho; Blos, Mauricio F. ; Junqueira, Fabricio ; Santos Filho, Diolino J. ; Miyagi, Paulo E. . **Control Architecture for Reconfigurable and Distributed Manufacturing System**. ABCM Symposium Series in Mechatronics - Part I - International Congress Section I - Modelling, Control & Identification. 1ed. Rio de Janeiro, 2014, v. 6, p. 219-230.
- (a) Este capítulo de livro é resultante do artigo enviado para o COBEM 2013. O artigo demonstra como deve ser desenvolvida uma arquitetura de controle para SRMs;
- (xi) Silva, R.M.; Junqueira, F.; Santos Filho, D.J.; Miyagi, P.E. **Design of reconfigurable and collaborative control system for productive systems**. ABCM Symposium Series in Mechatronics. Rio de Janeiro, 2012, v. 5, p. 868-878.
- (a) Este capítulo de livro é resultante do artigo enviado para o COBEM 2011. Aborda aspectos básicos do método proposto no Capítulo 4;

Como citado anteriormente, alguns trabalhos enviados para evento foram selecionados para serem publicados em periódicos ou capítulos de livros, e por já terem sido comentados não são relacionados aqui. Os demais trabalhos completos publicados em anais de eventos e decorrentes da pesquisa são:

- (xii) Silva, R.M., Santos Filho, D.J., Miyagi, P.E. **Modelagem de sistema de controle da Indústria 4.0 baseado em holon, agente, rede de Petri e arquitetura orientada a serviços**. Simpósio Brasileiro de Automação Inteligente (SBAI), 2015.
- (a) Este trabalho foca em demonstrar como a proposta apresenta características inovadoras e pode ser considerada um método para a concepção e análise de controle de sistemas de outros paradigmas que necessitam considerar reconfiguração em seus processos, tais como o sistema de controle da Indústria 4.0;

(xiii) da Silva, R. M.; Junqueira, Fabricio; Santos Filho, Diolino J. ; Miyagi, P.E. **Implementing Reconfigurable and Distributed Manufacturing Control System**. In: 23rd International Congress of Mechanical Engineering, COBEM 2015, 2015, Rio de Janeiro, RJ. Proceedings of COBEM 2015. Rio de Janeiro, RJ: ABCM, 2015.

(a) O trabalho apresentado no COBEM 2016 foca no desenvolvimento da implementação e desenvolvimento do exemplo de aplicação;

(xiv) da Silva, Robson Marinho; Junqueira, Fabricio ; Dos Santos Filho, Diolino J. ; Miyagi, Paulo E. . **A method to design a manufacturing control system considering flexible reconfiguration**. In: 2014 12th IEEE International Conference on Industrial Informatics (INDIN), 2014, Porto Alegre. p. 82-87. <http://dx.doi.org/10.1109/INDIN.2014.6945488>

(a) O trabalho apresentado no INDIN, além do método, explorou aspectos de SRMs, principalmente, a consideração da flexibilidade de reconfiguração, ou seja, a aplicação dos conceitos de reconfiguração tanto para desenvolver reações a ocorrência de falhas como a flexibilidade de mudanças de demanda.

(xv) Silva, R. M.; Junqueira, Fabrício ; Santos Filho, Diolino J. ; Moscato, L. A. ; Miyagi, Paulo E. . **Método para projeto de sistemas de controle supersório considerando flexibilidade de reconfiguração**. In: 20^o Congresso Brasileiro de Automática (CBA), 2014, Belo Horizonte, MG. Anais do CBA 2014. São Paulo, SP: Sociedade Brasileira de Automática (SBA), 2014. p. 1576-1583.

(a) Este trabalho apresentado no CBA2014 também explorou a consideração da flexibilidade de reconfiguração.

(xvi) Silva, R.M.; Junqueira, F.; Santos Filho, D.J.; Miyagi, P.E. **Modeling of holonic manufacturing control systems considering reconfiguration and timing constraints** . In: Anais do CBA, XIX Congresso Brasileiro de Automática, Campina Grande, PB, 2012.

(a) Aspectos do tratamento de restrições temporais para SRMs são abordados neste artigo. As principais considerações desenvolvidas foram incluídas no na Seção 3.4.

Os trabalhos seguintes abordam aspectos iniciais e foram elaborados durante o desenvolvimento do projeto de doutorado, ou seja, abordam os fundamentos que deram origem a solução proposta:

- (xvii) Silva, R.M.; Martins, M.P.; Junqueira, F.; Santos Filho, D.J.; Miyagi, P.E. **Projeto de sistema ativo de controle holônico para sistemas produtivos**. In: Anais do COBEF 6^o Congresso Brasileiro de Engenharia de Fabricação, Caxias do Sul, RS., 2011.
- (xviii) Silva, R.M.; Junqueira, F.; Santos Filho, D.J.; Miyagi, P.E. **Desing of control systems based on holon, Petri net and multi-agent system** . In: Anais do SBAI X Simpósio Brasileiro de Automação Inteligente, São João del-Rei, MG, 2011.
- (xix) Silva, R.M.; Cardoso, M.P.S.; Martins, M.P.; Junqueira, F.; Santos Filho, D.J.; Miyagi, P.E. **Design of Reconfigurable and Collaborative Control System for Productive Systems** . In: Proceedings of COBEM 21st International Congress of Mechanical Engineering, Natal, RN, 2011.