UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

**JOÃO ALVAREZ PEIXOTO**

# DESENVOLVIMENTO DE SISTEMAS DE AUTOMAÇÃO DA MANUFATURA USANDO ARQUITETURAS ORIENTADAS A SERVIÇO E SISTEMAS MULTI-AGENTES

Porto Alegre
2012

**JOÃO ALVAREZ PEIXOTO**

# DESENVOLVIMENTO DE SISTEMAS DE AUTOMAÇÃO DA MANUFATURA USANDO ARQUITETURAS ORIENTADAS A SERVIÇO E SISTEMAS MULTI-AGENTES

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.

Área de concentração: Controle e Automação.

ORIENTADOR: Prof. Dr. Carlos Eduardo Pereira

Porto Alegre

2012

JOÃO ALVAREZ PEIXOTO

# DESENVOLVIMENTO DE SISTEMAS DE AUTOMAÇÃO DA MANUFATURA USANDO ARQUITETURAS ORIENTADAS A SERVIÇO E SISTEMAS MULTI-AGENTES

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Carlos Eduardo Pereira - UFRGS

Doutor pela Stuttgart University - Stuttgart, Alemanha

Banca Examinadora:

Prof. Dr. José Antônio Barata de Oliveira, UNINOVA

Doutor pela Universidade Nova de Lisboa – Lisboa, Portugal

Prof. Dr. Marcelo Götz, UFRGS

Doutor pela Universidade Paderborn - Alemanha e UFRGS, Brasil

Prof. Dr. Renato Ventura Bayan Henriques, UFRGS

Doutor pela Universidade Federal de Minas Gerais, Brasil

Coordenador do PPGEE: _____

Prof. Dr. João Manoel Gomes da Silva Junior

Porto Alegre, agosto de 2012.

# DEDICATÓRIA

Dedico este trabalho a minha esposa Solange e meus filhos Taise e João, em especial pela compreensão e apoio em todos os momentos que necessitei me dedicar a este trabalho.

Mesmo com todo conhecimento que se possa alcançar, mesmo com toda vontade de aprender, mesmo com todo aprendizado que se possa ter, de nada vale se não tem com quem compartilhar. E quando tudo isto acabar, ainda resta a família que sempre esteve contigo.

# AGRADECIMENTOS

Ao Programa de Pós-Graduação em Engenharia Elétrica, PPGEE, pela oportunidade de realização de trabalhos em minha área de pesquisa.

Aos colegas do PPGEE pelo seu auxílio nas tarefas desenvolvidas durante o curso e apoio na revisão deste trabalho, em especial ao colega André Cavalcante que mesmo estando em Portugal me auxiliou a cada dúvida a que lhe recorria.

A EEP SENAI Ney Damasceno Ferreira, que propiciou horário e laboratórios para que as aulas e pesquisas ocorressem.

Aos professores Márcio Soares Torres (SENAI), Rubem Sprenger Dreger (UNISINOS) e Luciano Lopes Pfitscher (UNISINOS), por ter me referenciado ao ingresso no programa de mestrado da UFRGS.

Ao orientador Carlos Eduardo Pereira, por conduzir os caminhos deste trabalho, orientando e instigando na busca por um trabalho de caráter significativo.

A minha filha Taise e esposa Solange, que mesmo sem entender nada sobre agentes, se propuseram a ler e corrigir a dissertação inúmeras vezes.

A Deus por ter me concedido o dom de aprender.

# RESUMO

Os requisitos de manufatura industrial apontam uma necessidade de reconfiguração e reprogramação do fluxo de processo, a fim de atender modificações no produto com as mudanças dos requisitos de mercado. Estas mudanças implicam em alterações no processo de fabricação, o que, em muitos casos, significa alterar o leiaute, reprogramar controladores, modificar acoplamentos e interfaceamentos, etc. Os tamanhos dos lotes de produção estão ficando menores e a variedade de produtos em uma mesma linha tem aumentado. Isto torna o tempo de preparação dos dispositivos de manufatura fator crucial na formação do custo final. Dispositivos com mais autonomia, capazes de se autogerenciar e que permitam uma troca rápida de funcionalidades passam a serem desejados em um ambiente de manufatura, onde os sistemas convencionais com programação centralizada, sequência definida no controlador central e arranjo de funcionalidades fixas deixam a desejar. Orientação a serviços é um conceito que propõe a descentralização do controle e a disposição de componentes de forma a se agruparem virtualmente e formar novos dispositivos, com novas funcionalidades, otimizando investimento, recurso e tempo de produção. Nesta abordagem os equipamentos passam a se comportar como agentes de manufatura, em um universo de multi-agentes, que negociam entre si o processo requerido, propiciando o atendimento à variação de produto e dispondo de um número maior de funcionalidades, o que reduz o tempo de troca de processos. Um estudo de caso de um sistema orientado a serviços, sob uma plataforma multi-agentes, é proposto neste trabalho, buscando fornecer subsídios para a análise de desempenho e potencialidades deste conceito.

**Palavras-chave: Sistemas multi-agentes, arquitetura orientada a serviços, automação, manufatura industrial.**

# ABSTRACT

The industrial manufacturing requirements indicate a need for reconfiguration and reprogramming of process flow in order to meet changes in the product with the changing market requirements. These changes imply changes in the manufacturing process, which in many cases, means changing the layout, reprogram controllers, modify interfaces and couplings etc. The production lot sizes are getting smaller and the variety of products in the same line has increased. This makes the preparation time of manufacturing devices crucial factor in the formation of the final cost. Devices with more autonomy, able to manage themselves and allow a rapid exchange of features are to be desired in a manufacturing environment, where conventional systems with centralized scheduling, sequence defined in the central controller and array of features still fall short. Service orientation is a concept that proposes to decentralize the control and disposition of components in order to regroup virtually and form new devices with new features, optimizing investment, resource and production time. In this approach the devices start to act as agents of manufacturing, in a universe of multi-agents, who negotiate among themselves the process required, providing care for the product variation and having a greater number of features, which reduces the switching time processes. A case study of system-oriented services, under a multi-agent platform is proposed in this work, aiming to provide tools for performance analysis and potential of the concept.



**Keywords: Mult-agent System. Service Oriented Architecture. Automation. Industrial Manufacturing.**

# SUMÁRIO

# LISTA DE ILUSTRAÇÕES

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS

AID – Agent Identifiers Definition (Definição do Identificador do Agente)

AMS – Agent Management System (Gerenciamento do Sistema de Agentes)

CAD – Computer Aided Design (Desenho Assistido por Computador)

CAE – Computer Aided Engeneering (Engenharia Assistida por Computador)

CAM – Computer Aided Manufacturing (Manufatura Assistida por Computador)

CAPP – Computer Aided Process Planning (Planejamento do Processo Assistido por Computador)

CAQ – Computer Aided Quality (Qualidade Assistida por Computador)

CIM - Computer Integrated Manufacturing (Manufatura Integrada por Computador)

CLP – Programable Logic Control (Controlador Lógico Programável)

DF – Directory Facilitator (Postagem de Serviços)

DSC – Discentred System Control (Sistema de Controle Descentralizado)

DPWS – Device Profile for WEB Services (Perfil de Dispositivo para Serviços naWEB)

EAS - Evolvable Assembly System (Sistemas Evolutivos de Manufatura)

FBD - Function Blocks Diagram (Diagrama de Blocos Funcionais)

FIPA – Fundation for Intelligent Physical Agents (Fundação para Agentes Físicos)

FIPA-ACL - FIPA-Agent Communication Language (FIPA–Linguagem de Comunicação de Agentes)

FMS - Flexible Manufacturing System (Sistemas Flexíveis de Manufatura)

IEC – International Electrotechnical Commission (Comissão Internacional de Eletrotécnica)

IL - Instruction List (Lista de Instruções)

JADE – JAVA Agent Development Framework (Plataforma de Desenvolvimento de Agentes)

JVM – JAVA Virtual Machine (Máquina Virtual JAVA)

LD - Ladder Diagram (Diagrama Ladder)

MAS – Multi-Agent System (Sistema Multi-Agentes)

MMS – Manufacturing Messaging Services (Serviços de Mensagem para Manufatura)

PCP - Production Control Planing (Planejamento e Controle da Produção)

RMS – Reconfigurable Manufacturing System (Sistemas Reconfiguráveis de Manufatura)

SFC - Sequencial Function Chart (Sequenciamento Gráfico de Funções)

SOA – Service Oriented Architecture (Arquitetura Orientada a Serviços)

ST - Struct Text (Texto Estruturado)

# 1. INTRODUÇÃO

## 1.1 EVOLUÇÃO DOS REQUISITOS DA INDÚSTRIA DA MANUFATURA

A indústria de manufatura tem se deparado com uma necessidade de mercado que remete ao consumo de produtos altamente personalizados. Durante vários anos o conceito de produção em massa, caracterizada pela produção do mesmo produto em grande escala, foi amplamente implementado, mas hoje em dia é incapaz de tratar as variações do tipo de produto e não consegue mais responder aos desafios de modernidade e dinamismo. Grandes lotes de produção, linhas de produção com máquinas e processos idênticos e a padronização de produtos deixaram de existir. A produção em grandes quantidades continua a ser demandada, porém, como aponta (LOBOV et al., 2008), há uma tendência à produção em massa de produtos altamente personalizados.

Requisitos de qualidade e funcionalidade deixaram de ser os únicos atributos em um produto. A eles agregou-se diversidade, atualização, possibilidade de agregar funcionalidades, dentre outros. Requisitos que remetem a um sistema de produção cada vez mais ágil, flexível e eficaz. O estudo de (JAMES; SMIT, 2005) já apontava que o ambiente das empresas no futuro seria caracterizado por constantes mudanças na demanda do mercado e que a competitividade global pressionaria a entrada de novos produtos. E apontava também o dado de que um terço do custo total de uma fábrica, durante sua vida útil, é gasto na instalação e configuração de máquinas e equipamentos. A necessidade de minimização do tempo de inatividade de um sistema de manufatura é outro fator considerável, assim como a alteração do fluxo do processo para introduzir novos produtos ou para substituir antigo. Em (JAMES; SMIT, 2005) são definidos alguns dos requisitos da chamada fábrica do futuro: capacidade de interação dinâmica; cooperação entre as empresas; suporte de *hardware* heterogêneo

interoperáveis; escalabilidade, adicionando recursos sem interromper as operações; agilidade através da adaptabilidade e reconfiguração; e por fim, a tolerância à falha e recuperação dela.

Ao longo dos anos os sistemas de manufatura buscaram maximizar a produção e agora se deparam com estes novos requisitos. E este talvez seja o ponto da mudança de paradigma, de um sistema onde o controlador do processo era projetado para maximizar produção, para um sistema focado a dispor serviços eficazes, ágeis e flexíveis, maximizando a diversidade.

A concorrência mundial entre as empresas acelera a busca por atender este novo mercado, altamente diversificado. Surge então a necessidade de novos sistemas para realizar o controle da distribuição de produção, através da integração de sistemas de informação e recursos de auto-organização, adaptando-se rapidamente às mudanças no ambiente, como mostra (LEITÃO; RESTIVO, 2001).

## 1.2 NOVOS PARADIGMAS PARA SISTEMAS DE MANUFATURA

Na década de 80, as empresas japonesas já utilizavam o paradigma chamado de Produção Enxuta (*Lean Manufacturing*), vide (WOMACK, 2004), com o objetivo de redução do tempo decorrido entre o pedido de um cliente e o envio do produto , através da eliminação de resíduos, redução do tempo de projeto, menores estoques, menos defeitos, redução dos *setups*, etc. *Just in Time* é uma extensão do conceito de *Lean Manufacturing* (JÓZEFOWSKA, 2007), que consiste em ter o material certo, no lugar certo e no tempo certo, como aponta (LEITÃO et al., 2001). No conceito Just in Time já há a preocupação em reduzir o tempo de preparação dos equipamentos, em face de um aceno de que a variedade de produtos estava por aumentar. Até então o foco de uma empresa era a produção de um mesmo produto em larga escala. Quando havia mais de um produto a ser produzido pela empresa, esta então se dividia em minifábricas específicas por produtos, vinculadas à mesma empresa.

Do ponto de vista da produção, ainda era uma fábrica cuja vocação era a produção em massa de um determinado produto.

Junto ao *Lean Manufacturing*, surgiu outro paradigma semelhante que é a manufatura ágil (*Agile Manufacturing*), citado em (QUINN, 1997), definida como a capacidade de se adaptar rapidamente e lucrativamente às mudanças contínuas e inesperadas no ambiente de produção. No paradigma da Manufatura Integrada por Computador (CIM), que consiste na integração de todas as atividades da empresa por meio da utilização das tecnologias da informação e comunicação, como: bancos de dados, redes, protocolos de comunicação, entre outros, que permitem a troca e o compartilhamento de dados entre as unidades da empresa e diversas aplicações. A necessidade de agilidade na produção tornou-se mais evidente, assim como os requisitos de flexibilidade, em função do conceito de produzir mais de um produto na mesma linha de produção. Os Sistemas Flexíveis de Manufatura (FMS) (KUSIAK, 1986) são apontados como uma resposta à necessidade de médios volumes de produção, alta variedade do produto, mas com capacidade de produção menor do que em linhas de produção dedicadas. O objetivo é a customização em massa. A flexibilidade é alcançada com a previsão de todas as trocas na linha de produção, como relata (CAVALCANTE; PEREIRA; BARATA, 2010), mas onera a ociosidade de sistemas que não estão sendo utilizados em um determinado processo de produção, mas que estão ali em função de uma futura necessidade.

Desde então as necessidades de produção diversificadas aumentaram e as empresas de manufatura têm se adequado a essa nova realidade através da implantação de FMSs e leiautes que permitam que um sistema de montagem seja compartilhado por mais de um processo produtivo. A adaptação mais significativa da indústria a um mercado que demanda produção, agilidade e diversidade, diz respeito à otimização de leiautes. Os sistemas produtivos ainda utilizam conceitos de gerenciamento centralizado (processo centrado no CLP), manufatura integrada (CIM) e manufatura flexível ou montagem flexível (FMS/FAS). E isto de certa

forma tem atendido satisfatoriamente os requisitos de produção e diversidade, porém a agilidade ainda é um desejo não atendido.

Analisando o ciclo de vida de um produto, detendo-se nas etapas de desenvolvimento à produção, percebe-se a necessidade de desenvolvimento e produção de uma grande variedade de produtos, visando atender a um mercado ávido por inovações e serviços customizados que acabam implicando em um aumento na diversidade de produtos. Ciclos de produtos distintos necessitam ocorrer concomitantemente, como apontado em (OLIVEIRA, 2003). O desenvolvimento de um produto é definido pelas etapas de especificação, projeto e desenvolvimento, implementação e produção propriamente dita. Na etapa de especificação ocorre o levantamento de requisitos, a delimitação das especificações em função das funcionalidades requeridas e a viabilidade de produção.

O desenvolvimento do projeto diz respeito ao protótipo com as funcionalidades requeridas e seus métodos de fabricação. A implementação é o ensaio de produção, onde o produto é ensaiado no processo produtivo, sendo validados os processos de manufatura e seus parâmetros. Então o produto é colocado em marcha no processo produtivo, com os parâmetros, requisitos e insumos definidos no plano de processo. Nota-se então que a etapa de implementação requer os recursos de manufatura, tanto quanto a produção propriamente dita, mas com o foco em protótipo. Em um sistema produtivo moderno, é comum que no momento que um produto entra no ciclo de produção, um novo produto já deve estar em fase de implementação. Isto exige um sistema de manufatura que possa ser reconfigurado para atender aos dois processos. A Figura 1 ilustra graficamente os ciclos de três produtos sendo desenvolvidos concomitantemente, atendendo à diversidade exigida pelo mercado.

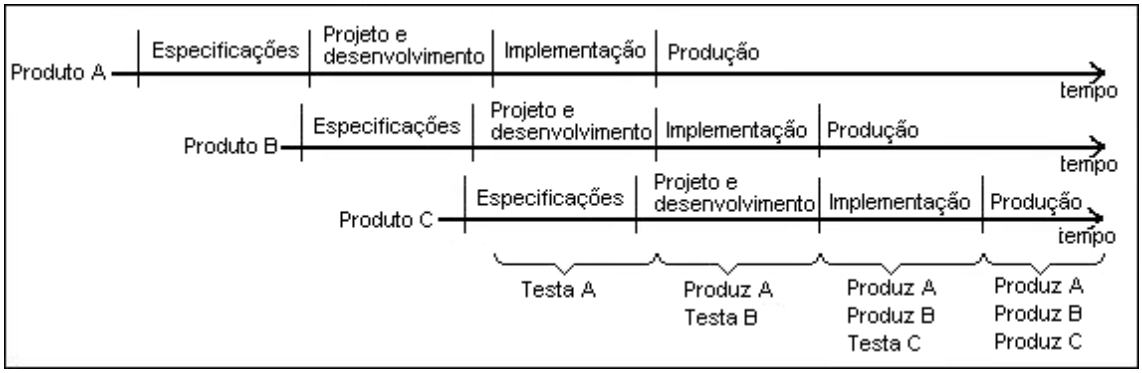


Figura 1: Sobreposição no ciclo de vida de produtos.

Segundo (JAMES; SMIT, 2005), os principais requisitos que a próxima geração de sistemas de produção deve ter são: integração empresarial, arquiteturas organizacionais distribuídas, ambientes heterogêneos, integração dos seres humanos com *hardware* e *software*, cooperação, estrutura dinâmica e aberta e estrutura da organização tolerante a falhas O desenvolvimento de aplicações de fabricação que atendam a esses requisitos e apoiem as novas estruturas organizacionais apresentam alguns problemas importantes, que não estão completamente resolvidos, tais como: compartilhamento de dados, planificação e programação do processo produtivo e técnicas de reorganização. Os novos sistemas de controle para este novo ambiente devem impor uma capacidade de adaptação ágil e rápida às mudanças dos ambientes, através da integração das novas tecnologias, ferramentas e paradigmas.

Os sistemas evolutivos de manufatura (EAS) (CAVALCANTE; PEREIRA; BARATA, 2010), com suas funcionalidades de autoconfiguração, reorganização, autonomia de gerenciamento do processo produtivo, disponibilidade de serviços ao sistema (utilizando conceitos de “arquiteturas orientadas a serviço” ou em inglês “service oriented architectures - SOA) e disposição para negociação e agendamento de processos entre seus agentes (usando conceitos de “sistemas multi-agentes” ou em inglês – “multi-agent systems”), se apresentam como uma tecnologia promissora para atender aos requisitos de mercado de produtividade em

massa e diversidade de produção, dispondo da flexibilidade de um FMS e a autonomia de um sistema descentralizado (DSC), como consta em (LEITÃO; RESTIVO, 2001). Todavia, é ainda pequeno o número de implementações de sistemas evolutivos de manufatura e são poucas as avaliações quantitativas, baseadas em processos reais ou em simulações, com indicadores do desempenho destes sistemas. Este será o objetivo deste trabalho.

## 1.3 OBJETIVOS

O trabalho de dissertação tem o objetivo de propor uma abordagem de gerenciamento de sistemas de manufatura no conceito de arquitetura orientada a serviços, valendo-se das funcionalidades de sistemas multi-agentes.

Especificamente este trabalho trata de uma analise do estado da arte sobre pesquisas que propõem o uso de MAS e SOA em manufatura; propõe a abordagem que combine de um sistema de manufatura MAS e SOA; implementa e valida da proposta em um sistema flexível de manufatura e compara o desempenho da proposta com métodos "tradicionais“, determinando através de métricas o desempenho do sistema.

## 1.4 ORGANIZAÇÃO DO TRABALHO

O Capítulo 2 apresenta uma revisão literária dos sistemas de gerenciamento da manufatura.

O Capítulo 3 resgata os conceitos de tecnologias que dão suporte a sistemas de manufatura evolutivas, abordando tópicos como “arquiteturas orientadas a serviços” – SOA (do inglês “service-oriented architectures”), sistemas multi-agentes – MAS (do inglês “multi-agent systems”), linguagem JAVA, plataforma JADE e padrão FIPA-ACL de comunicação entre agentes.

O Capítulo 4 apresenta a análise do estado da arte de trabalhos relacionados.

O Capítulo 5 apresenta a proposta conceitual desta dissertação e aplica o método utilizado para implementações em sistemas de manufatura.

O Capítulo 6 traz a implementação realizada do sistema de manufatura utilizando SOA/MAS, a planta didática virtual com uso da plataforma JADE e comunicação utilizando o padrão FIPA-ACL, que será realizado para validar a proposta conceitual e comparar com um sistema implementado utilizando a norma IEC61131.

O Capítulo 7 apresenta a validação experimental através de um estudo de caso, onde são apresentados os resultados da análise de desempenho do sistema e uma correlação entre as propostas convencionais de sistemas de manufatura e a proposta de utilização de orientação a serviços em sistemas multi-agentes.

O Capítulo de conclusão relaciona os resultados da proposta e aponta para possíveis trabalhos futuros dentro da linha de pesquisa de sistemas de manufatura evolutivos, autogerenciáveis e reconfiguráveis.

# 2. GERENCIAMENTO DE SISTEMAS DE MANUFATURA

O gerenciamento dos sistemas de manufatura é um importante tema de estudos, visto que o tipo de sistema é que vai descrever as características da manufatura, seja flexibilidade de processo, confiabilidade, desempenho, agilidade, produção, e outras. Estas características são desejadas pelo mercado produtivo, o que remete a novos estudos e ensaios de novos conceitos.

## 2.1 CONTROLE CENTRALIZADO DOS PROCESSOS

Tradicionalmente o controle em um sistema de automação é realizado de forma centralizada e hierárquica, como aponta (COLOMBO et al., 2008). Um componente capaz de processar informações é encarregado de receber dados dos dispositivos de entrada (sensores, chaves, teclados, etc.) e acionar os dispositivos de saída (atuadores como eletroválvulas, motores, solenóides, etc.), segundo uma relação lógica e previamente definida em um programa de automação.

Um dos componentes programáveis mais utilizado no controle de sistemas de automação é o Controlador Lógico Programável (CLP). Um Controlador Lógico Programável é um componente que concentra todos os pontos de entradas e saídas de sinais, que controla o fluxo do processo e gerencia o processo de manufatura. Conforme (GEORGINI, 2000) o CLP inspeciona os sinais de todas as entradas, executa o processo lógico pré-definido, ajusta o estado lógico das saídas, conforme a lógica do processo. Isto se repete em ciclos. A Figura 2 mostra a estrutura do sistema com gerenciamento centrado no CLP, onde os sinais de todos os componentes do sistema são conectados neste controlador central.

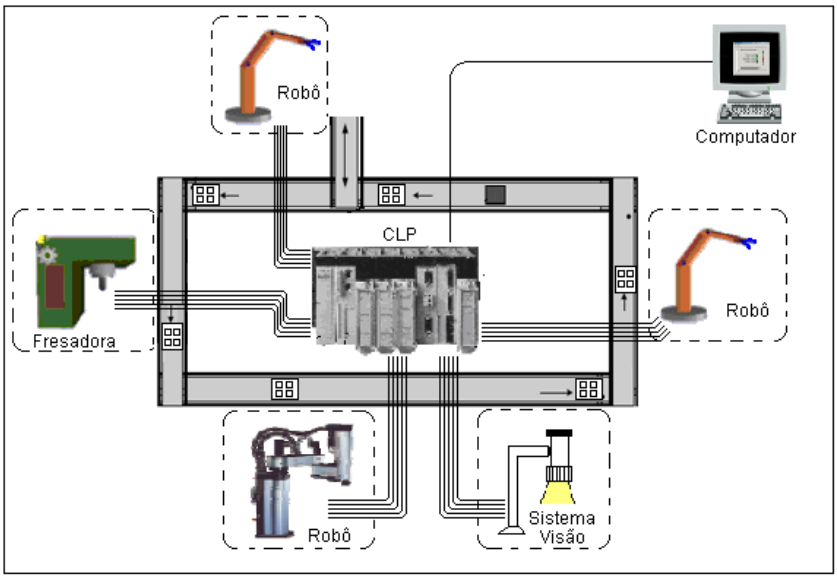


Figura 2: Gerenciamento centrado no CLP.

Os controladores lógicos programáveis têm sua linguagem de programação especificada pela norma IEC61131 (*International Electrotechnical Commission* - IEC), que tem sua versão corrente publicada em 2003. Essa norma possui 8 partes, onde cada parte especifica um tópico relacionado aos controladores lógicos:

a) Parte 1: Definição da informação geral, da terminologia básica e dos conceitos;

b) Parte 2: Exigências de equipamento e testes eletrônicos e testes mecânicos de construção e verificação;

c) Parte 3: Estrutura do *Software* do CLP, execução do programa e linguagens de programação;

d) Parte 4: Guia de orientação ao usuário na seleção, instalação e manutenção de CLPs;

e) Parte 5: Especificação de mensagens de comunicações baseadas em MMS (*Manufacturing Messaging Services*);

f) Parte 6: Comunicação de CLPs utilizando IEC *fieldbus*;

g) Parte 7: Programação utilizando Lógica *Fuzzy*;

h) Parte 8: Guia para implementação das linguagens.

A norma IEC 61131-3, parte 3, define 5 linguagens de programação, como relata (GUIMARÃES, 2005), que são:

a) Texto Estruturado (ST – do inglês “Structured Text”);
b) Lista de Instruções (IL – do inglês “Instruction List”);
c) Diagrama de Blocos Funcionais (FBD – do inglês “Function Block Diagram”);
d) Diagrama de Relés (LD – do inglês “Ladder Diagram”);
e) Sequenciamento Gráfico de Funções (SFC – do inglês “Sequential Function Charts”).

Texto Estruturado é uma linguagem de fácil entendimento para os programadores com conhecimentos de algoritmos e linguagens de programação procedurais, pois possui identificadores de fácil entendimento e fácil interpretação. Com a facilidade de digitação direta de fórmulas, esta linguagem é muito útil para desenvolvimento de cálculos aritméticos. Sua programação se dá pela descrição em linhas de comandos/instruções, encadeados entre diretivas.

Já o Diagrama de Blocos Funcionais é uma linguagem gráfica. Blocos funcionais, que representam operações lógicas ou aritméticas, são representados em retângulos (blocos), onde as relações lógicas entre entradas e saídas são representadas por linhas de interconexão.

Um Diagrama de Relés ou Ladder é uma linguagem gráfica baseada nos diagramas elétricos, onde os sinais de entrada e as relações lógicas são representadas como contatos elétricos abertos ou fechados, e os sinais de saída e componentes internos funcionais (contadores, temporizadores, relés auxiliares, etc.) são representados como bobinas elétricas. As interconexões entre os componentes são realizadas através da representação de linhas, com a intenção de simbolizar a fiação elétrica. Uma linha vertical à esquerda representa um barramento energizado e à direita uma barra de terra, com o fluxo de potência sempre da

esquerda para a direita. A função de controle é definida pela forma com que os contatos (abertos ou fechados) são associados para comandar a bobina do relé (série ou paralelo). É a linguagem mais familiar ao eletricista, em face de similaridade aos diagramas elétricos multifilares dos sistemas de comando elétrico.

A linguagem de programação por Lista de Instruções (*Instruction List* - IL) é uma linguagem textual, de baixo nível, com estrutura semelhante à linguagens de máquina ou assembler, ideal para resolver problemas simples e pequenos, onde existem poucas quebras no fluxo de execução do programa e onde a otimização dos programas do ponto de vista de redução do código gerado e desempenho de execução são importantes.

Por fim, o sequenciamento Gráfico de Funções (*Sequential Function Chart* - SFC) é uma linguagem composta por uma série de passos (etapas) mostrados como retângulos conectados por linhas verticais. Cada retângulo representa um passo na execução de uma atividade, podendo também ser associado ao conceito de estado. Entre cada retângulo (estado) há uma linha horizontal que representa uma transição entre um estado e outro. Esta transição está relacionada a uma condição de entrada e a uma condição de sequência desejada. Quando a condição da transição for verdadeira então o fluxo passa de um estado para outro. Esta linguagem comporta que seus blocos de estados sejam programados em outras linguagens, apresentando portabilidade entre elas.

Uma vez que o sistema de gerenciamento da manufatura com o controle centralizado implica que os sinais de entrada e saída do sistema se concentrem no componente de gerenciamento (CLP como exemplo), não é difícil imaginar que, com o aumento da quantidade de sinais a serem processados, a quantidade de cabos de alimentação e transmissão de sinais também se eleva. Surgem então as “redes industriais”, que além de propciar o sincronismo entre os dispositivos, viabilizam o intercâmbio de informações entre os diversos componentes de um sistema de automação. Em um sistema de automação sempre

encontramos elementos sensores, controladores e atuadores, e na maioria das vezes interfaces homem máquina ou mesmo sistemas de supervisão para permitir a interação entre o operador e o sistema. A comunicação entre esses elementos é essencial, e as redes de comunicação industriais implementam essa comunicação. Mesmo com a intenção de padronização dos protocolos através da norma IEC61158, não há um consenso entre os grandes fabricantes e comissões, o que remete a uma gama de redes industriais disponíveis a implementar comunicação entre dispositivos e um CLP. As redes industriais resolvem os problemas de grande número de conexões dos sistemas de gerenciamento centrados no CLP.

A facilidade de programação e visão do gerenciamento do processo são os pontos positivos deste sistema de gerenciamento da manufatura, porém, a reconfiguração de parte do sistema usualmente significa ter que reprogramar tudo: *software*, *hardware* e ligações elétricas, além de que a falha de um componente afeta todo o sistema. A diversividade de produtos é um problema que remete a reprogramação quase que total do sistema.

## 2.2 MANUFATURA INTEGRADA POR COMPUTADOR - CIM

Com a visão sobre o processo de manufatura, entende-se um CIM como a união dos dispositivos que possuem um gerenciamento local, interligados a um computador central que gerencia o fluxo do processo e os algoritmos de controle executados em cada máquina, como mostra a Figura 3.

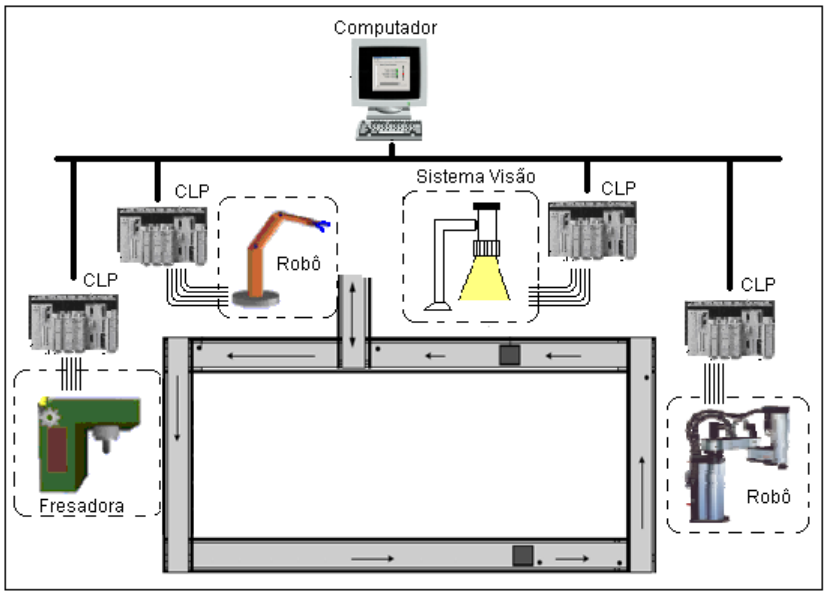


Figura 3: Sistema de Manufatura Integrada por Computador – CIM.

Mas isto é a aplicação do CIM no chão de fábrica. O conceito de CIM é mais amplo e visa à integração entre todas as etapas do processo: vendas, suprimentos, projeto e desenvolvimento, produção, expedição e pós-vendas. Segundo (LEITÃO et al., 2001), o paradigma CIM consiste na integração de todas as atividades da empresa por meio da utilização das tecnologias da informação, como: bancos de dados, redes, etc, que permitem a troca e o compartilhamento de dados entre as unidades da empresa e suas aplicações. Para (FLEISCHHAUER, 1996) a manufatura integrada por computador é o eficiente uso da tecnologia de informação em manufatura para aumentar a produtividade e eficiência de empresas.

Numa visão global e genérica, pode-se estabelecer que uma Manufatura Integrada por Computador tem seu início com o planejamento da produção (projeto do produto, do processo e da estimativa de quantidades a produzir), continua com a programação (definição precisa de produtos a produzir no período, cálculo de necessidades de material, estabelecimento de prazos e capacidades e sequenciamento), aciona a produção e termina no controle, através de

módulos de captação de dados de quantidade e qualidade da produção, havendo, ainda, a possibilidade de trabalhar com funções de controle de qualidade.

Assim, um CIM é composto pelo Planejamento e Controle da Produção (PCP), pela Engenharia Assistida por Computador (CAE), pelo Projeto Assistido por Computador (CAD), pelo Planejamento do Processo Assistido por Computador (CAPP), pela Produção Assistida por Computador (CAM) e pela garantia de Qualidade Assistida por Computador (CAQ). A Figura 4 apresenta os componentes de um CIM.

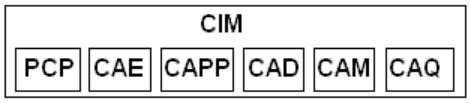


Figura 4: Componentes de um CIM.

A Engenharia Assistida por Computador (CAE) baseia-se na construção e teste de protótipos em nível de *software*, onde se simula a resistência dos materiais, por exemplo, através da variação de temperatura e força, reduzindo dessa forma os custos e tempo de projeto, e ao mesmo tempo em que se aprimora a qualidade do produto. Isso acarreta um sensível ganho de tempo no desenvolvimento, levando à vantagem competitiva decorrente do lançamento de produtos mais rapidamente.

O Projeto Assistido por Computador (CAD) são as atividades de projeto que envolve o uso do computador para criar, modificar ou documentar um projeto de engenharia. O CAD não deve ser visto como instrumento restrito ao projeto ou desenho, sendo uma forma de integração entre projetistas e respectivas equipes e demais setores da empresa, inclusive clientes e fornecedores. Através do CAD pode-se criar e manipular desenhos em um sistema de coordenadas cartesianas, alterando suas propriedades. Já em três dimensões, pode-se atribuir noção de volume a objetos construídos a partir de formas sólidas, tais como cubo,

cilindro, esfera, pirâmide, entre outras, além de possibilitar o cálculo de perímetros, áreas, volumes, fazer simulações e desenhos.

Já o Planejamento do Processo Assistido por Computador (CAPP) tem a função de estabelecer o roteiro ou o processo de fabricação de um produto. É a determinação sistemática dos métodos, através dos quais um produto deve ser fabricado. Trata-se de determinar e selecionar máquinas, ferramentas, instruções de trabalho e demais condições necessárias à transformação dos pedidos em produtos finais. No CAPP são listados os processos que são capazes de serem realizados, a sequência de operação que o produto vai seguir, a distribuição do trabalho pelas máquinas, o cálculo dos tempos de fabricação, o dimensionamento das sobras de materiais e a programação das máquinas para a execução do processo estabelecido.

A Manufatura Assistida por Computador (CAM) é a aplicação da informática e da tecnologia de informação e comunicações ao sistema de produção, no sentido de eliminar a inconstância e perda de tempo inerentes à manipulação e decisão do ser humano. O CAM, enquanto auxílio à produção, pode se restringir ao controle ou ser extensivo ao planejamento, e ainda auxiliar na monitoração dos recursos de produção.

O sistema de Qualidade Assistida por Computador (CAQ) constitui-se de um acompanhamento desde a chegada dos insumos, passando pelo processo produtivo, estendendo-se até a saída do produto acabado. Este acompanhamento garante a qualidade do produto através de seus pontos de inspeção e permite que um processo de manufatura possa ser monitorado desde as suas etapas iniciais.

Os módulos que compõem um CIM interagem, permitindo a troca de informações do processo de manufatura programado. O sistema de produção inicia-se pela elaboração do projeto mediante o auxílio de sistemas CAE e CAD. Pelo PCP tem-se a geração de lista de materiais e seus custos. O CAPP trata-se do roteiro de produção. E por fim o CAM

implementa a manufatura, segundo informações armazenadas em seu banco de dados relativas ao produto.

Um sistema integrado de manufatura traz o controle sobre o processo bem definido e a possibilidade de variação do produto no mesmo sistema de controle da produção, uma vez definida no CAPP. Suas estações são autônomas, o que permite que o sistema continue em funcionamento mesmo com a retirada de uma estação.

Algumas características levantadas por (BUSSMANN, 1998), como: serem flexíveis, possuir em uma hierarquia de controle fixo, a reconfiguração e ampliação dos sistemas existentes são difíceis de serem alcançadas no CIM. O desempenho da produção não é mantido quando o sistema está fora das condições normais. Mas a capacidade de integrar todas as partes de um projeto, da sua concepção a sua manufatura, leva o CIM a um conceito de manufatura amplamente aceito no meio industrial.

A integração total do gerenciamento da produção passa a ser um ganho quando se tem todas as etapas interligadas, desde o suprimento ao fornecimento, incluíndo o planejamento de produção, planejamento dos processos de produção, o controle de qualidade, e outros. Mas a divesidade de produção ainda é um problema, pois mesmo que se tenha o controle do gerenciamento, a troca da produção requer alterações no fluxo do processo, intervenção nas máquinas e reprogramação.

## 2.3 SISTEMAS FLEXÍVEIS DE MANUFATURA - FMS

A necessidade de diversidade de produção impõe limites no tempo necessário para adaptar as máquinas ou equipamentos de um produto a outro. Alterar o produto fabricado implica normalmente na modificação do seu sistema produtivo, efetuando os ajustes necessários para a nova fabricação. Os sistemas flexíveis de manufatura vêm apresentar a possibilidade de um equipamento realizar mais de uma tarefa. Para que o mesmo equipamento

possa realizar mais de uma operação na manufatura, o mesmo deve ser dotado de recursos que permitam, através de dispositivo de alimentação, alterar suas funcionalidades, proporcionando processos distintos sendo realizado pelo mesmo equipamento. Um exemplo de sistema flexível de manufatura pode ser visto na Figura 5.

Figura 5: Composição de um FMS (FESTO, 2011).

Segundo consta em (CAVALCANTE; PEREIRA; BARATA, 2010) um FMS se distingue de outras formas de manufatura automatizada por considerar a diversidade dos produtos que se deseja produzir (flexibilidade do produto) e as características adaptativas das máquinas (flexibilidade dos equipamentos).

Sistemas de informações para o planejamento, sequenciamento e coordenação das operações de forma integrada são implementados através do uso de tecnologias da informação, utilizadas no controle dos equipamentos de produção. Desta forma (BARROS, 2006) define um FMS como uma combinação de equipamentos, sistemas de controle e comunicação integrados na manufatura, visando um desempenho de alta produtividade, capacidade de respostas de modo rápido e econômico a mudanças no ambiente operacional.

Um FMS, entre outras formas de manufatura automatizada, se distingue por considerar: a diversidade de produtos, as características adaptativas das máquinas e as propriedades de similaridade dos processos. Para atender a divesidade de produtos, as máquinas e equipamentos que compõe o FMS são dotados de funcionalidades que permitem realizar mais de uma operação, sendo as máquinas passíveis de adaptação aos processos requisitados dentro de uma similaridade.

Os volumes de produção de um sistema flexível de manufatura são normalmente baixos, mas com uma grande diversidade de produtos distintos. Uma vez que o FMS possui funcionalidades distintas agregadas, durante a produção somente uma das funcionalidades estará em uso, ficando as demais funcionalidades ociosas. Por isto um alto volume de produção não se justifica neste sistema, pois a capacidade de ser flexível se destina a grandes quantidades de troca de produção, o que implicará em baixo volume de produção por lote.

O aumento do nível de utilização dos equipamentos, a melhora do nível de qualidade do produto, a redução do custo de produção e dos tempos de preparação da máquina (*setup*) e o rastreamento dos produtos ao longo da produção são algumas das vantagens que um sistema flexível de manufatura apresenta.

Estes sistemas impõem um problema desafiador, que se traduz na correta alocação dos recursos aos diversos processos requeridos para produzir uma gama de produtos, assim como a programação da sequência das atividades para se obter o melhor desempenho do sistema.

A diversividade de produção neste sistema é atendida, porém a ociosidade de recursos devido à flexibilidade e ao uso de um recurso por vez é o ponto deficitário do sistema de produção em FMS.

## 2.4 SISTEMA DE MANUFATURA BIÔNICA - BMS

Sistema de produção inspirado em sistemas biológicos é uma abordagem que faz um paralelo entre as propriedades de sistemas biológicos e sistemas de manufatura. O conceito foi criado por (UEDA, 1992a), que observou que os organismos biológicos podem se adaptar, autoreorganizar, autoexpandir e evoluir. Este conceito foi desenvolvido durante a década de 90, visando que características dos sistemas biológicos, tais como autonomia e comportamento espontâneo, fossem copiados para sistemas de manufatura. O sistema é visto como a combinação de vários sistemas autônomos, relacionados hierarquicamente, que se combinam para formar uma determinada função (THARUMARAJAH; WELLS; NEMES, 1998).

Em outro trabalho (UEDA, 1992b) mostra que a produção pode ser vista como um sistema dividido em sistemas pequenos, serviços autônomos que interagem e compartilham para formar uma tarefa maior. As interações são alcançadas através da troca de informações DNA-tipo (hereditária) e BN-tipo (processo de aprendizagem). De fato, o fluxo de informação material (DNA-tipo), juntamente com o fluxo de trabalho (ADN-tipo) entre as entidades de produção (NB-tipo) define um produto (BN + DNA ) e fazem o ajuste das informações e do fluxo, onde o sistema pode reagir às mudanças no ambiente. O processo de fazer esse ajuste é conhecido como *síntese problema*. As semelhanças entre os sistemas de produção e sistemas biológicos são também abordadas em (GU et al., 2011). As células autônomas que trocam informações para sintetizar órgãos podem ser vistas como simples agentes autônomos.

## 2.5 SISTEMA DE MANUFATURA HOLÔNICA - HMS

Outra abordagem para sistemas de manufatura é o desenvolvimento dos chamados Sistemas de Manufatura Holônicos (HMS). Esta visão é baseada no conceito de *holon*.

Segundo (BUSSMANN, 1998) a aplicação dos conceitos holônicos na fabricação foi inicialmente motivada pela incapacidade dos sistemas de produção existentes em lidar com a evolução dos produtos dentro de uma unidade de produção. Os *holons* cooperam entre si para atingir as metas de produção e este processo de cooperação também envolve humanos.

Segundo o Filósofo Arthur Koestler em seu livro "The ghost in the machine", *HOLOS* equivale a "todo" e *ON* equivale a "em", assim, *HOLON* explica a evolução dos sistemas biológicos e sociais.

Um *holon* é um todo em uma abordagem parte, (XUEMEI et al., 2004). HMS tem como princípios: um sistema é um todo formado por partes; sistemas complexos irão evoluir a partir de sistemas simples. Em (CAVALCANTE; PEREIRA; BARATA, 2010) este conceito de que sistemas complexos são contituídos de sistemas simples é reforçado.

A abordagem HMS permite adaptabilidade por meio da agregação de diferentes níveis de *holons* ou pela especialização. No entanto, a preocupação é dissociar a estrutura do sistema do algoritmo de controle, permitindo a gestão da complexidade e uma reprogramação rápida, caso necessário. Mas não é uma iniciativa de auto-organização real, porque o paradigma não aborda os mecanismos de reconfiguração. Alguma flexibilidade é possível por meio da cooperação entre os *holons*, mas não há definição formal de como o sistema vai evoluir se algumas mudanças substanciais ocorrem no sistema.

Um *holon* sempre contém uma parte de processamento de informação e, opcionalmente, uma transformação física.

## 2.6 SISTEMAS DE MONTAGEM EVOLUTIVOS – EAS

A tendência do mercado consumidor a novas chamadas para a produção em massa de produtos altamente personalizados requer que máquinas e equipamentos possam ser mutáveis, segundo especificidade do processo de produção momentâneo. Muitos autores têm

identificado o conceito de Sistemas de Montagem Evolutivos (EAS) como uma forma de responder a esse cenário.

Os conceitos atuais são baseados na necessidade de se fazer alterações em um sistema sempre que o ambiente muda. Em (BARATA et al., 2001) descobre-se que sistema de montagem evolutivo é baseado em sistemas simples, reconfiguráveis, elementos com tarefas específicas (módulos do sistema), que permitem uma evolução contínua do sistema de montagem. Mais recentemente (FERREIRA et al., 2009) apresentou uma proposta na qual um Sistema de Montagem Evolutivo (EAS) pode co-evoluir juntamente com o produto e o processo de montagem.

Em estudo na literatura, pode-se definir EAS como um sistema de montagem dinâmico, auto-organizado e evolutivo, formado por módulos interligados e capaz de mudar sua estrutura de acordo com as mudanças ambientais relevantes, ou com a capacidade de atender as flutuações de demanda de processos produtivos (CAVALCANTE; PEREIRA; BARATA, 2010).

Os sistemas de montagem evolutivos estão mais próximos do gerenciamento do processo distribuído, onde cada equipamento de montagem tem seu gerenciamento próprio, capaz de interagir, apresentar suas funcionalidades e de reconhecer demandas de funcionalidades que requerem uma reconfiguração (evolução) do sistema de montagem. Diz-se que o fluxo de processo passa a ser gerenciado pelo palete.

No sistema EAS, segundo (CAVALCANTE; PEREIRA; BARATA, 2010), se um módulo falha, alguma funcionalidade pode ser perdida, mas o sistema pode se ajustar e formar novas coalizões ou diferentes formatações de constituição, suprindo o módulo perdido. Assim, o sistema pode apresentar propriedades que não são encontradas em partes. Estas propriedades ou comportamentos surgem para auto-organizar o sistema. O EAS é formado por uma coalizão de diversos pequenos componentes. Cada componente é visto como um

agente e o sistema como multi-agente, com habilidade de auto-organizar-se. O tempo de reconfiguração é reduzido para próximo de zero e aumenta a capacidade de produzir uma variedade maior de produtos.

Surge então a figura dos agentes mecatrônicos (MA), que, segundo (CAVALCANTE, 2012), podem ser:

a) Agente de Recursos (RA): são os agentes vinculados aos recursos físicos e que oferecem serviço;
b) Agente de Produtos (PA): são os agentes vinculados aos produtos, os quais solicitam serviços;
c) Agente de transporte (TUA): cuja função é transportar os PAs dentro do sistema.

A auto-organização se dá respeitando as três fases: solicitação de alocação do recurso, execução e desalocação do recurso. A proposta final é a mudança rápida de manufatura, de forma que uma produção em marcha pode auto-organizar-se de forma a absorver um protótipo. Assim, o tempo de desenvolvimento e teste de um produto cai, reduzindo o seu tempo de chegada ao mercado ("*time-to-market*").

A evolução dos sistemas atende a necessidade de produtos diversificados, altamente personalizados e que necessitam de funcionalidades de manufatura que no primeiro momento os elementos talvez não as tenham, mas que podem evoluir (através de coligações) para atendê-las.

## 2.7 SISTEMAS RECONFIGURÁVEIS DE MANUFATURA - RMS

Junto ao EAS há também o conceito de Sistemas de Manufatura Reconfigurável (RMS), visto que é a possibilidade de reconfiguração que torna o sistema de montagem

evolutivo. RMS é citato por (COLOMBO et al., 2008) como sistemas integrados por equipamentos que permitam sistemas com modularidade, adaptabilidade e escalabilidade do produto. Mas para serem totalmente operacionais, tais módulos devem oferecer *hardware* aberto e arquiteturas de *software* com funcionalidade *plug and play*, ou seja, sua conexão é realizada sem que os demais equipamentos sejam desligados ou reconfigurados.

Os sistemas reconfiguráveis permitem uma adaptação tanto de *hardware* como de software a uma necessidade específica de produção. Requisitos de modularidade faz com que os RMSs agreguem módulos para formar funcionalidades compostas, como uma coalizão entre estruturas, compondo uma funcionalidade que o sistema individualmente não tinha. O processo de reconfiguração deve onerar o mínimo de tempo possível, a fim de que a nova funcionalidade possa ser disposta à manufatura sem que haja prejuízos na produção.

Aqui têm-se a funcionalidade da adaptação rápida a mudança, através da reconfiguração. Um novo produto com novos requisitos de manufatura pode ter suas etapas de manufatura definidas pela reconfiguração dos equipamentos de forma a atendê-lo.

# 3. TECNOLOGIAS DE SUPORTE AO DESENVOLVIMENTO DE SISTEMAS DE MANUFATURA

Para o desenvolvimento de sistemas de manufatura com os conceitos de arquitetura orientada a serviços e utilizando sistemas multi-agentes recursos tecnológicos se fazem necessários. As ferramentas que gerenciam os sistemas ditos tradicionais não dão conta das funcionalidades requeridas nestes novos conceitos. Novas ferramentas são utilizadas em auxílio ao desenvolvedor, com funcionalidades definidas e classes abertas para que o desenvolvedor crie seus próprios códigos.

## 3.1 AGENTE

A FIPA (*Foundation for Intelligent Physical Agents*) define agente como uma entidade que reside em um ambiente onde interpretam dados através de sensores, refletem eventos no ambiente e executam ações que produzem efeitos no ambiente. Um agente pode ser *software* puro ou *hardware* e *software* compostos.

No geral, agentes são entidades de *software* autônomas que atuam em determinado ambientes de forma a interagir com este e com outros agentes. Além de produzir ações e percepções sem requerer intervenções humanas constantes.

Em (SILVA, 2003) tem-se uma classificação dos tipos de Agentes, como segue:

a) Agentes móveis: são agentes que têm a mobilidade como característica principal, isto é, uma capacidade de mover-se por uma rede interna local (intranet) ou até mesmo pela WEB;
b) Agentes situados ou estacionários: são aqueles opostos aos móveis. São fixos em um mesmo ambiente ou plataforma;

c) Agentes competitivos: são agentes que “competem” entre si para a realização de seus objetivos ou tarefas. Não há colaboração entre os agentes;

d) Agentes coordenados ou colaborativos: agentes com a finalidade de alcançar um objetivo maior realizam tarefas específicas, porém coordenando-as entre si de forma que suas atividades se completem;

e) Agentes reativos: são agentes que reagem a estímulos sem ter memória do que já foi realizado no passado e nem previsão da ação a ser tomada no futuro;

f) Agentes Cognitivos: ao contrário dos agentes reativos, eles podem raciocinar sobre as ações tomadas no passado e planejar ações a serem tomadas no futuro.

Tem-se o agente como um sistema capaz de não somente agir autonomamente, mas também, em alguns casos, interagir com outros agentes, os quais, assim como ele, objetivam alcançar suas metas. Um agente pode ainda ter algumas características que o tornem mais ou menos útil para uma determinada tarefa, tais como: autonomia, interatividade, adaptabilidade, sociabilidade, mobilidade, representatividade, pró-atividade, inteligência, racionalidade, imprevisibilidade, continuidade temporal, transparência, coordenação, cooperação, competição, robustez e confiabilidade.

A arquitetura de uma proposta de agentes utilizados em manufatura, segundo (LEITÃO; RESTIVO, 2001), é baseada em quatro módulos e um banco de dados local, que contém todas as informações pertinentes sobre o comportamento do agente, demonstrado na Figura 6.

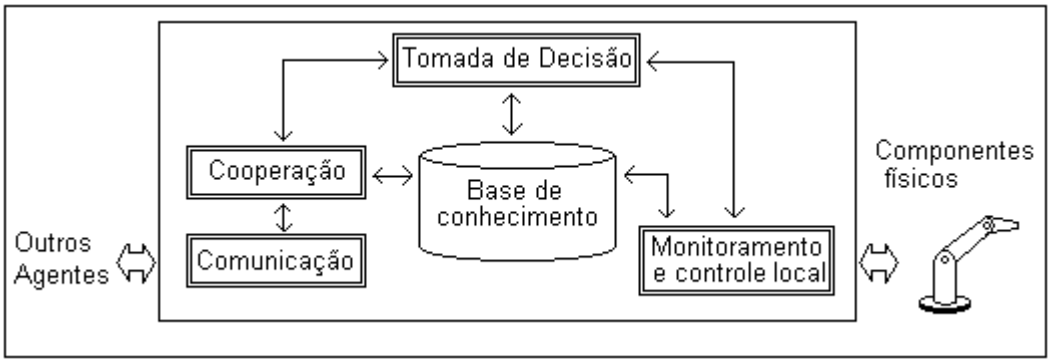


Figura 6: Composição de um agente utilizado em manufatura.

Pode-se descrever cada módulo como:

a) Módulo tomada de decisão: este módulo controla todas as atividades do agente e inclui a resolução de problemas e tomada de decisão;
b) Módulo de cooperação: este módulo gerencia a cooperação com agentes exteriores, solicitando a cooperação de outros agentes, coletando as respostas e enviando para o módulo tomada de decisão;
c) Módulo de comunicação: este módulo trata da necessidade de padronizar a interação entre agentes distribuídos e define uma comunicação linguística;
d) Módulo de monitoramento e controle local: este módulo pretende controlar e monitorar a execução operacional do agente;
e) Módulo de base de conhecimento: o banco de dados armazena todos os conhecimentos sobre o comportamento do agente.

## 3.2 ARQUITETURA ORIENTADA A SERVIÇOS - SOA E SISTEMAS MULTI-AGENTES - MAS

Empurrado pela instabilidade do mercado e pela turbulência, as empresas modernas são levadas a adotar metodologias inovadoras de negócios para ganhar flexibilidade e agilidade, como mostra (BARATA; MENDES; RIBEIRO, 2008). A arquitetura orientada a

serviços e os sistemas multi-agentes se propõem a dar conta da agilidade e flexibilidade requerida. Em referenciais teóricos encontram-se diversos artigos que apontam soluções aos problemas de automação utilizando os princípios de SOA e outros artigos apontam MAS como instrumento. A dúvida que surge diz respeito à diferença entre MAS e SOA.

O Sistema Multi-Agente (MAS) é uma tecnologia de *software* que é capaz de modelar e implementar comportamentos individuais e sociais em sistemas distribuídos. No conceito MAS os sistemas, agora chamados de agentes, são ativos, ou seja, não apenas podem postar seus serviços e se submetem a serem solicitados em suas funcionalidades, mas assumem papel proativo para iniciar comunicação com outros agentes, propor negociação e alocar serviços.

Segundo (JAMES; SMIT, 2005) SOA é o conjunto de princípios arquitetônicos para a construção autônoma de sistemas interoperáveis. Eles são criados independente uns dos outros e operam independentemente do seu meio, fornecendo funcionalidades próprias (podem funcionar sozinhos). Esta autonomia é ressaltada por (BARATA; RIBEIRO; COLOMBO, 2007) que relaciona SOA a um conjunto de arquitetura para a construção autônoma de sistemas interoperáveis, que suportam sistemas autônomos e oferecem funcionalidades de autosuficiência, funcionamento independente e flexibilidade.

A arquitetura orientada a serviços é um conceito onde um processo pode ser visto como um conjunto de sistemas que dispõem serviços que se complementam para formar um processo produtivo. Cada sistema dentro de um SOA presta serviços independente dos demais e ao serem conectados ao conjunto eles dispõem seus serviços aos outros sistemas. Uma vez solicitados, os sistemas de manufatura são alocados e dispõem seus serviços. A Figura 7 mostra um sistema de manufatura segundo o principio de SOA.

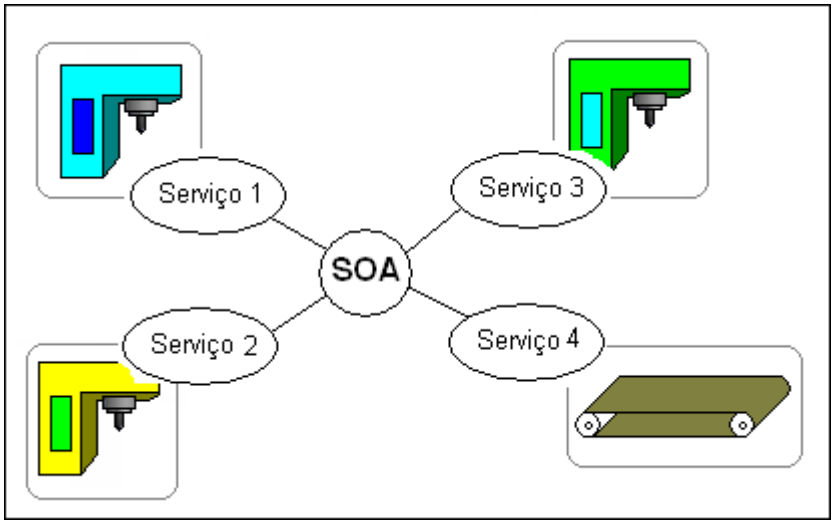


Figura 7: Exemplo de um SOA.

Neste exemplo, os serviços 1, 2, 3 e 4 são dispostos uns aos outros e cada um deles vale-se de tarefas a que estão vinculadas. Um serviço pode implementar mais de uma tarefa, ou mesmo compor duas ou mais tarefas para formar um serviço único.

Fica claro no exemplo que cada sistema é autônomo e que o gerenciamento do fluxo de processo só depende da disponibilidade dos serviços, podendo outros sistemas ser agregados ou retirados, em função da necessidade de funcionalidades diversificadas. Mas o sincronismo deste gerenciamento não é trivial. (COLOMBO et al., 2008) já apontava o desafio de como descrever os processos que regulamentam a comportamento do sistema e como sincronizar e coordenar a execução dos serviços oferecidos pelas entidades distribuídas para atingir o comportamento desejado.

Assim, SOA e MAS se complementam. Em ambos casos cada participante no processo de montagem (robô, pinça, transportador, armazém de ferramentas, etc) é identificado como um agente que pode oferecer diversos serviços, e que interagem na realização de tarefas cooperativas.

Em (BARATA; MENDES; RIBEIRO, 2008), um estudo comparativo entre os conceitos de SOA e MAS é realizado, sendo apontadas algumas diferenças e semelhanças

entre os conceitos, assim como sua complementaridade. Ele aponta algumas características do SOA, tais como:

a) Autonomia: não há nenhuma dependência direta entre os serviços e que são estruturalmente dissociados;
b) Interoperabilidade: é conseguido com especificação de uma interface que descreve os serviços que estão sendo acolhidos e os padrões de interação considerados;
c) Independência de plataforma: idealmente os serviços são descritos por meio de formatos baseados em texto. Essas representações não são ligadas a uma arquitetura de computador, sistemas operacionais, linguagem de programação ou de tecnologia e pode ser facilmente decodificado por qualquer sistema;
d) Encapsulamento: serviços proporcionam funcionalidades autocontidas, que são expostas por interfaces definidas, escondendo detalhes desnecessários;
e) Disponibilidade: os serviços podem ser publicados nos registros públicos e disponibilizados para uso geral;
f) *WEB Services*: é o mecanismo preferido para implementação de SOA.

E neste mesmo estudo aponta características dos sistemas multi-agentes, tais como:

i. Autonomia: um agente é autônomo quando é capaz de agir sozinho, sem a ajuda de terceiros (como outros agentes ou seres humanos);
ii. Sociabilidade: um agente deve ser capaz de se comunicar com outros agentes ou mesmo outras entidades;
iii. Racionalidade: um agente pode raciocinar sobre os dados que recebe, a fim de encontrar a melhor solução para alcançar seu objetivo;
iv. Reatividade: um agente pode reagir sobre as mudanças no ambiente, alterando seu comportamento em conformidade;

v. Proatividade: um agente proativo tem algum controle sobre suas reações, baseando-se em sua própria agenda e objetivos;

vi. Adaptabilidade: um agente é capaz de aprender com as alterações do seu comportamento, quando uma melhor solução é descoberta, adaptando-se às mudanças no ambiente.

Enquanto o SOA enfatiza descrições baseadas em serviços hospedados e não fornece um modelo de referência de programação, e que é normalmente suportado por tecnologias WEB utilizadas, garantindo a interoperabilidade com uma ampla gama de sistemas, no MAS o apoio se dá a métodos bem estabelecidos para descrever o comportamento de um agente. O fato de que os agentes são regulados por normas internas que apoiem a implementação do comportamento social é uma clara vantagem. Isto é de grande importância quando se considera sistemas que sofrem mudanças de execução dinâmica. MAS são otimizados para uso em LAN, são restritos ao cumprimento bem definido, mas com menos padrões de interoperabilidade utilizados. A tabela 1 apresenta uma comparação entre os sistemas SOA e MAS, relatada em (BARATA; MENDES; RIBEIRO, 2008).

Tabela 1 Análise comparativa entre SOA e MAS

| Característica | SOA | MAS |
|---|---|---|
| Unidade básica | Serviço. | Agente. |
| Autonomia | Ambas as entidades denotam autonomia já que a funcionalidade fornecida é autossuficiente | |
| Descrição do comportamento | Em SOA o foco está no detalhamento da interface pública, ao invés de descrever detalhes de execução. | Existem métodos bem estabelecidos para descrever o comportamento de um agente. |
| Habilidade social | Habilidade social não é definida para SOA, no entanto, a utilização de um serviço implica a aceitação das regras definidas na descrição de interface. | Os agentes denotam a capacidade social regulada por normas internas ou ambientais. |
| Complexidade do encapsulamento | A natureza autossuficiente das funcionalidades previstas permite esconder os detalhes. Em SOA este encapsulamento é explícito. | |
| Infraestrutura de comunicação | SOA são suportados pela WEB e tecnologias relacionadas que podem perfeitamente funcionar na internet. | As maiorias das implementações são otimizadas para uso de rede local. |
| Suporte para arquiteturas dinamicamente reconfiguráveis em tempo de execução | Reconfiguração muitas vezes exige uma reprogramação. | A natureza flexível dos agentes torna reativo às mudanças no ambiente. |
| Interoperabilidade | Assegurada pela utilização das tecnologias da WEB de propósito geral. | Fortemente dependente do cumprimento padrões FIPA. |
| Requisito computacional | Implementações leves com DPWS tem garantia de alto desempenho, sem restrições de interoperabilidade. | A maioria das implementações têm pesados requisitos computacionais. |

## 3.3 PLATAFORMA – JADE

JADE (*Java Agent Development Framework*) (SILVA, 2003) é uma plataforma implementada em linguagem JAVA. Ela simplifica a implementação de sistemas multi-agente através de um *firmware* que cumpre com as especificações FIPA (*Fundation for Inteligent Physical Agents*) e através de um conjunto de ferramentas gráficas que suporta as fases de

depuração e implantação. A plataforma de agentes pode ser distribuída através das máquinas (sem a necessidade de compartilhar o mesmo sistema operacional) e a configuração pode ser controlada através de uma interface gráfica remota. A configuração pode ser alterada, mesmo em tempo de execução por agentes do movimento de uma máquina para outra, como e quando necessário.

O objetivo do JADE é simplificar o desenvolvimento de sistemas multi-agentes, assegurando o cumprimento padrão através de um conjunto abrangente de serviços e agentes do sistema em conformidade com as especificações FIPA: serviço de nomes e serviços de páginas amarelas; transporte de mensagens e serviços de análise e uma biblioteca de protocolos de interação FIPA pronto para ser usado. A Figura 8 apresenta o ambiente JADE, na forma gráfica, que auxilia na monitoração dos componentes que estão na plataforma.

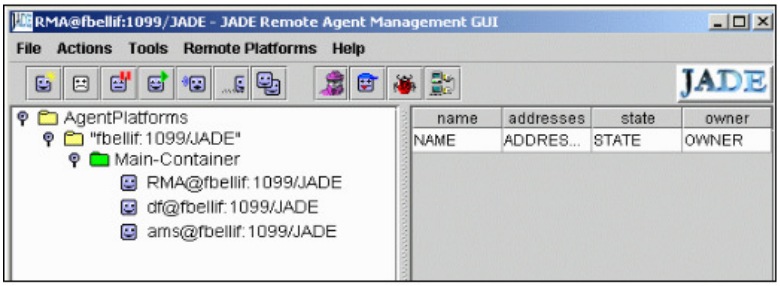


Figura 8: Ambiente JADE (TILAB, 2011).

Ao instalar o JADE, este já dispõe os recursos de interconectividade, bastando as máquinas estarem ligadas em rede, comunicando por TCP/IP. Um ambiente de execução onde os agentes JADE "vivem" e que devem estar ativos em um *host* antes que um agente possa ser executado. Junto a este *host* e a este agente são criados outros dois agentes (AMS e DF), sendo descritas suas funções a seguir:

a) Agente: é uma instância da classe *Agent* que representa a base para a definição de agentes. Já oferece todas as interações básicas da plataforma (registro, configuração,

etc.) e oferece um conjunto de métodos para a implementação do comportamento do agente JADE. No modelo computacional, um agente é um recurso multitarefa, onde os serviços são executados concorrentemente.

b) *Agent Management System* (AMS): agente que exerce o controle sobre o acesso e o uso da plataforma e mantém a lista de identificadores dos agentes (AID) que estão na plataforma. Todo agente deve se registrar no AMS;

c) *Directory Facilitator* (DF): oferece o serviço de páginas amarelas na plataforma, onde os demais agentes postam suas habilidades e onde os agentes podem buscar o endereço de agentes que possuem habilidades requeridas.

Somente o *host* possui, além do agente, o AMS e o DF. Os demais agentes serão executados em "*containers*", dentro da plataforma, podendo haver mais de um agente por *container*. A Figura 9 mostra um exemplo de estrutura de uma plataforma JADE.

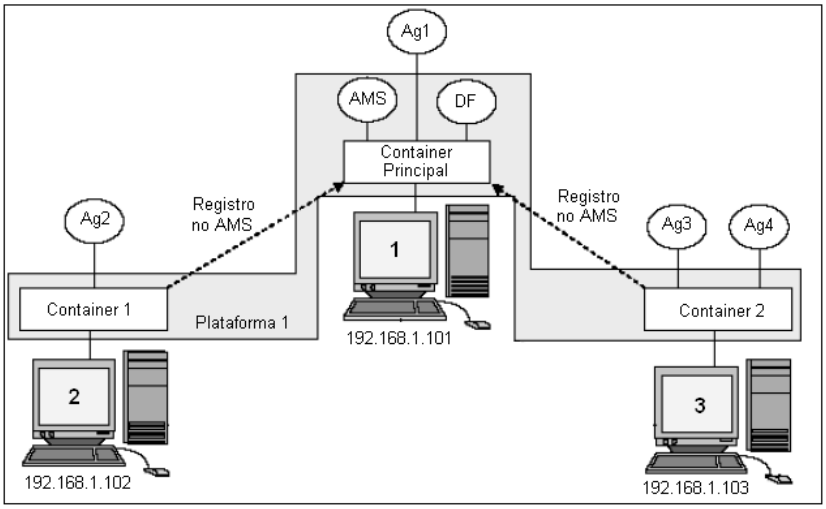


Figura 9: Exemplo de estrutura da plataforma JADE.

### 3.3.1 Agente em JADE

Do ponto de vista de programação, um agente JADE é uma instância da classe *Agent*, no qual os programadores ou desenvolvedores deverão escrever seus próprios agentes como

subclasses de *Agent*, adicionando comportamentos específicos de acordo com a necessidade e objetivo da aplicação, por meio de um conjunto básico de métodos e utilizando as capacidades herdadas que a classe *Agent* dispõe, tais como mecanismos básicos de interação com a plataforma de agentes (registro, configuração, gerenciamento remoto, etc.). A estrutura básica de um agente programado em JAVA é apresentado na Figura 10.

```
import jade.core.Agent;
public class AgenteTeste extends Agent {
    protected void setup() {
            System.out.println("Sou o agente " + getAID().getName());
            // inserir as demais tarefas.
            doDelete();
    }
    protected void takeDown() {
            System.out.println("Agente " + getAID().getName() + "terminando");
    }
}
```

Figura 10: Agente programado em JAVA.

O agente quando instanciado deve informar o identificador do agente (AID), sua classe de instância, antecedidos pelos comandos de execução, respeitando uma sintaxe. Para o exemplo da Figura 10 os comandos de instância para execução dos quatro agentes na plataforma 1 seria:

a) computador 1: >java jade.Boot –gui Ag1:nome_da_classe

b) computador 2: >java jade.Boot –host 192.168.1.101 –container Ag2:nome_da_classe

c) computador 3: >java jade.Boot –host 192.168.1.101 –container Ag3:nome_da_classe

d) computador 4: >java jade.Boot –host 192.168.1.101 –container Ag4:nome_da_classe

Descrevendo o significado de cada comando tem-se:

a) <java> comando para execução da JVM no computador;

b) <jade.Boot> programa a ser executado pela JVM;

c) <-gui> executa o *host* para hospedar dos agentes e o AMS e DF;

d) <-host numero_ip> identifica onde está localizado o *host* da plataforma;

e) <-container> cria um ambiente para hospedar um agente que não está no *host*;

f) <nome_agente> o *nome_agente* é um identificador que será atribuído a este agente e que será postado no AMS, uma vez postado no AMS, lhe será agregado o endereço origem (nome_agente@numero_ip:1099/JADE);

<nome_da_classe> o *nome_da_classe* identifica qual classe está sendo instanciada. Esta classe estende uma classe específica chamada *agent,* que possui as funcionalidades para implementação de agentes.

Quando executado o agente pelo comando *">java jade.Boot –gui Ag1:AgenteTeste",* o método *setup()* será executado, e é neste ponto em que as tarefas (métodos) deverão ser inseridas. Um agente, mesmo que não esteja executando algo, estará sempre ativo. Para encerrar o agente o método *doDelete()* deve ser invocado, que chamará o método *takeDown()*, encerrando o agente.

Um agente pode agregar em seu programa comportamentos específicos (*behaviours).* Os comportamentos são métodos cíclicos de execução, onde são descritas as ações que devem ser executadas, e são criados através do método *addBehaviour(new nome_classe_comportamento).* Comportamentos são descritos como sub-classes que estendem a classe *Behaviour*, e nelas são descritos dois métodos principais, que são:

a) *void action()*: descreve as ações a serem executadas em um ciclo;

b) *boolean done():* método que, se retornar *true* encerra a execução do comportamento, se retornar *false* coloca novamente o comportamento na fila de execução.

Quando um comportamento é adicionado, ele é encaminhado para uma fila de comportamentos em execução. Ao chegar sua vez, ele é executado e ao chegar ao final da execução o método *done( )* é inspecionado. Se ele retornar *false*, o comportamento se mantém na fila para ser executado na próxima vez que o fluxo de processo passar por ele. Caso o método retorne *true*, então é deslocado para a pilha de comportamentos encerrados. A Figura 11 demonstra este processo.

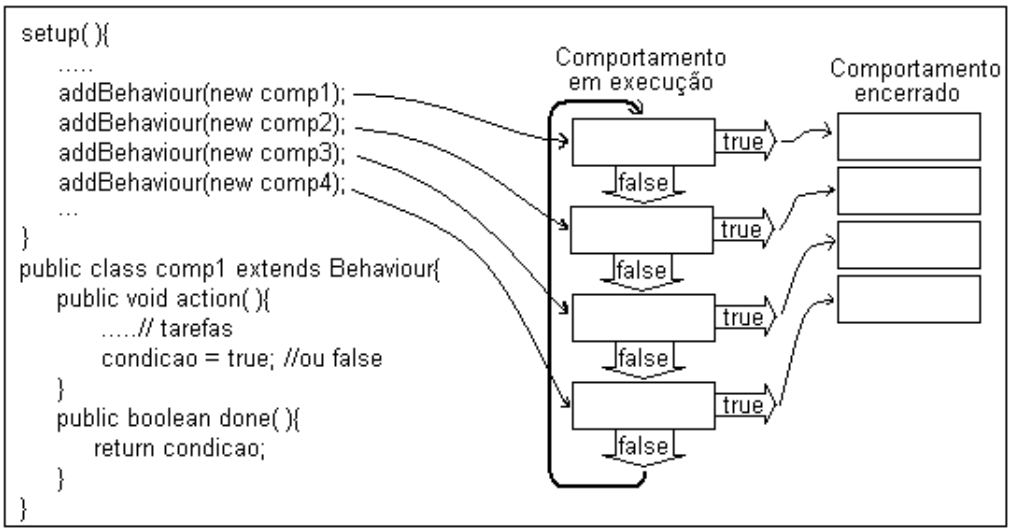


Figura 11: Formato de execução dos comportamentos no JADE.

Na Figura 12 está representado o fluxograma de execução de um agente. Nota-se que um comportamento é executado por vez, dentro do método de *setup().*

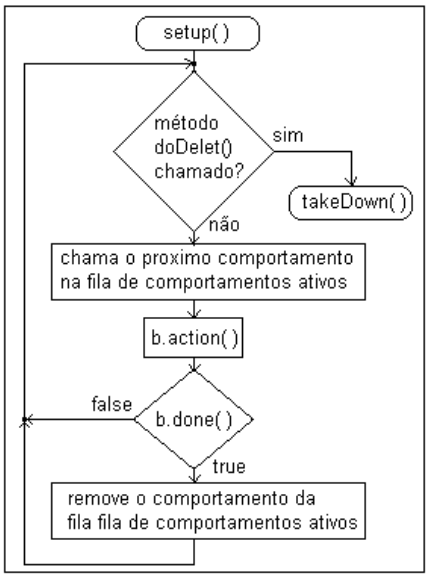


Figura 12: Fluxograma de execução de um agente.

Na implementação dos comportamentos tem-se a classe *Behaviour* que é uma classe abstrata do JADE, disponível no pacote *jade.core.behaviours.* Uma classe abstrata é uma classe que possui alguns métodos implementados e outros não, e não pode ser instanciada diretamente. Ela tem como finalidade prover a estrutura de comportamentos para os agentes JADE os implementarem. Nela encontram-se disponíveis comportamentos com funcionalidades próprias, onde se destacam os seguintes:

a) *SimpleBehaviour*: classe que estende a classe *Behaviour* (classe *jade.core.behaviours.Behaviour*), composta dos métodos *action()*, que promove as chamadas de tarefas, e *done()* que define se o comportamento irá continuar ou será encerrado;

b) *OneShotBehaviour*: classe que estende a classe *OneShotBehaviour* (classe *jade.core.behaviours.OneShotBehaviour*), composta somente do método *action()*. O método *done()* sempre retorna *true*, fazendo com que o comportamento seja executado uma única vez;

c) *CyclicBehaviour*: classe que estende a classe *CyclicBehaviour* (classe *jade.core.behaviours.CyclicBehaviour*), composta somente pelo método *action().* O método *done()* sempre retorna *false*, fazendo com que o seja repetido sempre que o *setup()* chamar a execução do comportamento;

d) *WakerBehaviour*: classe que estende a classe *WakerBehaviour* (classe *jade.core.behaviours.WakerBehaviour*), comportamento que espera um determinado período de tempo (em milisegundos) para efetivamente executar a tarefa;

e) *TickerBehaviour*: classe que estende a classe *TickerBehaviour* (classe *jade.core.behaviours.TickerBehaviour*), comportamento que executa uma tarefa periodicamente em intervalos de tempo constantes (em milisegundos). Este comportamento nunca acaba, pois seu método *done()* retorna sempre *false*.

### 3.3.2 Troca de mensagens entre agentes usando FIPA-ACL

A *Foundation for Intelligent Physical Agents* (FIPA) foi criada em 1996 por uma associação internacional sem fins lucrativos para desenvolver uma coleção de normas relativas à tecnologia de agentes de *software*. As composições iniciais, um grupo de acadêmicos e organizações industriais, elaboraram um conjunto de estatutos para orientar a criação de um conjunto de especificações padrão para tecnologias de agentes de *software*. Naquela época os agentes de *software* já estavam muito bem conhecidos na comunidade acadêmica, mas, até a data, só recebeu atenção limitada de empresas comerciais, além de uma perspectiva exploratória. O consórcio concordou em produzir normas que formariam a base de uma nova indústria, sendo utilizáveis através de um vasto número de aplicações.

O padrão FIPA é subdividido em um conjunto de 25 especificações. Além dessas especificações, deve fornecer uma linguagem para a comunicação entre os agentes com

rigorosas fórmulas semânticas. FIPA *Agent Communication Language* (FIPA-ACL) compreende cerca de 20 tipos de comunicação e todos eles são baseados na forma como se comunicam os seres humanos. Basicamente, trata-se de um formato onde uma mensagem ou comando é enviada/recebida por um agente. Isto permite que dois agentes, mesmo que com natureza distinta (plataforma, protocolo, camada física, entre outras), possam se comunicar.

Segundo consta em (BELLIFEMINE, 2004) o padrão FIPA encapsula uma mensagem, sendo dividido em camadas, em que cada uma destas camadas é responsável por uma parte no processo de comunicação. As subcamadas do padrão FIPA são:

a) Subcamada 1 (Transportes): responsável pelo protocolo de transporte. FIPA definiu protocolos de transporte de mensagens para IIOP, WAP e HTTP;
b) Subcamada 2 (Codificação): ao invés de enviar simples *byte* em mensagens codificadas, FIPA define várias representações de mensagens para o uso de estruturas de nível mais elevado de dados, incluindo XML *String*, e-Bit Eficiente. O último é destinado a ser utilizado quando comunicando através de conexões em baixa largura de banda;
c) Subcamada 3 (Conteúdo de mensagens): a estrutura da mensagem é especificada independente da codificação particular, para incentivar a flexibilidade;
d) Subcamada 4 (Ontologia): o termo individual contido na carga ou no conteúdo de uma mensagem FIPA pode ser explicitamente referenciado a um modelo de aplicação específica conceitual ou ontologia;
e) Subcamada 5 (Conteúdo): o teor real de mensagens FIPA pode ser de qualquer forma, mas FIPA definiu orientações para a utilização de fórmulas gerais lógicas e operações algébricas. A linguagem mais usada para expressar conteúdo é FIPA-SL;
f) Subcamada 6 (Ato comunicativo): a classificação de uma mensagem simples em termos de ação ou ato performativo[1]. Exemplos incluem informar, solicitar ou concordar;

[1] Diz-se de um enunciado que se dá ao mesmo tempo em que a ação por ele apresentada; palavra e ato coincidem.

g) Subcamada 7 (Protocolo de interação): raramente as mensagens são trocadas de forma isolada, mas sim fazem parte de alguma sequência de interação. FIPA define protocolos de interação, especificando as sequências de mensagens, como pedido que descreva uma parte de outro pedido, que por sua vez devem concordar ou recusar-se a cumprir.

A Linguagem ACL, que agrega ao padrão FIPA a troca de mensagens, é assíncrona e usa alguns campos principais, como citados a seguir:

a) *Sender*: AID do agente que envia a mensagem;
b) *Receivers*: lista de AID dos agentes destinos da mensagem;
c) *Performative*: ato de fala que indica o que o agente que envia a mensagem espera com dela;
d) *Content*: o real conteúdo da mensagem;
e) *Language*: sintaxe usada para expressar o conteúdo;
f) *Ontology*: denota a semântica dos itens do conteúdo;
g) Outros campos de controle de conversação, como: *conversation-id*, *reply-with*, *in-reply-to*, *reply-by*.

Toda a troca de mensagens realizada no JADE é feita através de métodos próprios e com o uso de instâncias da classe *ACLMessage*. Esta classe possui um conjunto de atributos que estão em conformidade com as especificações FIPA, implementando o padrão FIPA-ACL. Assim, um agente que pretenda enviar uma mensagem deve instanciar um objeto da classe *ACLMessage*, preenchê-los com as informações necessárias e chamar o método *send*() da classe *Agent*. Caso for receber mensagens, o método *receive*() da classe *Agent* deve ser chamado.

Para preencher as informações necessárias na instância de um objeto *ACLMessage*, os métodos seguintes devem ser utilizados:

a) *public void addReceiver(AID idAgente):* adiciona o AID de um agente como receptor ou destinatário da mensagem. Determina quem receberá a mensagem;

*b) public void removeReceiver(AID idAgente):* remove o AID do agente da lista de receptores;

c) *public ACLMessage createReply*(): cria uma nova mensagem ACL de resposta à determinada mensagem. Assim, o programador só necessita definir o ato comunicativo (*communicate-act*) e o conteúdo da mensagem;

d) *public static String[] getAllPerformativeNames():* retorna uma lista de *strings* com a lista de todos os atos performativos;

e) *public Iterator getAllReceiver():* retorna um objeto *iterator* contendo todos os AIDs dos agentes receptores da mensagem;

f) *public String getContent():* retorna uma *string* contendo o conteúdo da mensagem;

g) *public void setContent(String* conteúdo*):* define o conteúdo da mensagem a ser enviada;

h) *public void setOntology (String* ontologia*):* define a ontologia da mensagem.

A Figura 13 apresenta um exemplo de envio de mensagem FIPA-ACL com JADE. A Figura 14 apresenta um exemplo de recepção de mensagem FIPA-ACL com JADE.

```
public class AgenteEnvia extends Agent{
    protected void setup( ){
        ACLMessage msg = new ACLMessage( ACLMessage.INFORM );
        msg.addReceiver( new AID( "AgenteRecebe", AID.ISLOCALNAME ) );
        msg.setContent( "Oi amigo agente!" );
        send(msg);
    }
}
```

Figura 13: Exemplo de envio de mensagem FIPA-ACL.

```
public class AgenteRecebe extends Agent{
    protected void setup( ){
        addBehaviour(new ReceberMsg( ));
    }
    public class ReceberMsg extends CyclicBehaviour{
        public void action() {
            MessageTemplate mt;
            mt = MessageTemplate.MatchPerformative(ACLMessage.INFORM);
            ACLMessage msg = myAgent.receive(mt);
            if(msg != null) {
                System.out.println(msg.getContent( ));
            } else block( );
        }
    }
}
```

Figura 14: Exemplo de recepção de mensagens FIPA-ACL.

Nota-se aqui que a recepção fora implementada como um comportamento cíclico, para que ocorra a cada vez que uma mensagem chegar. Também cabe salientar o método *block()* que bloqueia o teste condicional até que uma mensagem chegue.

### 3.3.3 Protocolos especificados para troca de mensagens

Outro meio de enviar ou receber mensagens no JADE é através do uso das classes de comportamentos *SenderBehaviour* e *ReceiveBehaviour*. Fato que torna possível que as trocas de mensagens possam ser escalonadas como atividades independentes de um agente.

As interações se dão de forma estanque, onde cada método é executado segundo o ato performativo da mensagem que chega. Estes atos podem ser descritos, segundo *FIPA-ACL Communicative Act Library Specification SC00037*, como:

a) *Accept Proposal* – indica o aceite de uma proposta de uma mensagem anterior;

b) *Agree* – indica a concordância com uma ação proposta;

c) *Cancel* – indica a um agente que não tem a intenção de executar alguma ação;

d) *Call for Proposal* – indica a um agente uma proposta para realização de uma tarefa, com um convite;

e) *Confirm* – indica a um agente a confirmação de algo que foi proposto;

f) *Disconfirm* – indica a um agente que o que for proposto não está confirmado;

g) *Failure* – indica que a ação proposta por um agente falhou;

h) *Inform* – indica uma mensagem de informação a outro agente;

i) *Not Understood* – indica que uma mensagem recebida não foi devidamente entendida;

j) *Propagate* – indica a um agente que uma mensagem deve ser propagada a outro agente;

k) *Propose* – indica que está sendo emitida uma proposta para realização de uma ação;

l) *Refuse* – indica que uma proposta de ação foi abandonada;

m) *Reject Proposal* – indica que uma proposta de ação foi rejeitada;

n) *Request* – indica a requisição de alguma ação a um agente;

o) *Subscribe* – indica uma notificação a um agente.

A Figura 15 apresenta a estrutura do protocolo FIPA-ACL especificado por SC00037, em que cada método é executado de forma independente, segundo o ato performativo da mensagem que é lida.

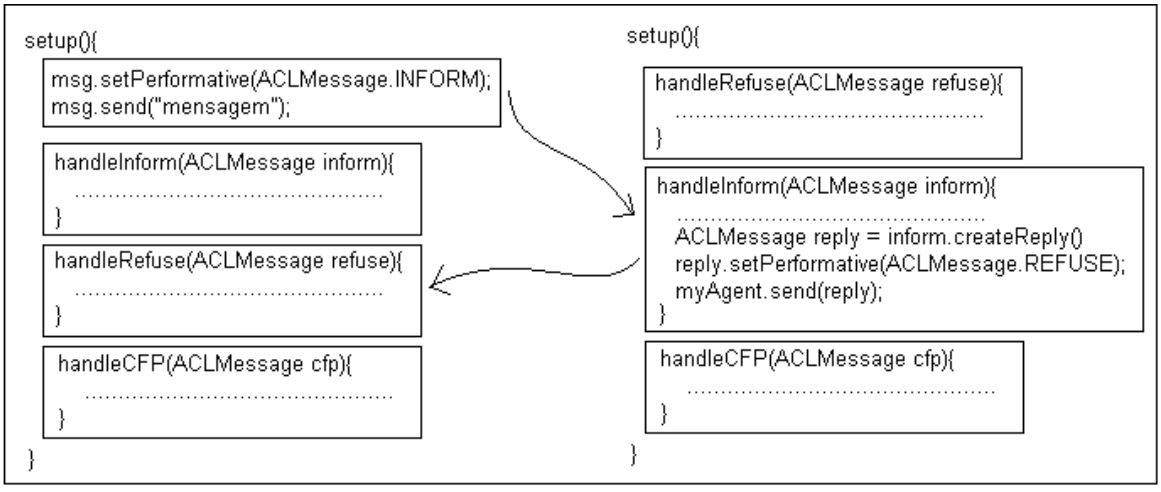


Figura 15: Estrutura do protocolo FIPA-ACL.

Há alguns protocolos descritos no padrão FIPA, sendo que os mais usuais são os protocolos *FIPA Request Interaction Protocol Specification (SC00026)* e *FIPA Contract Net Interaction Protocol Specification (SC00026*).

3.3.3.1 Protocolo FIPA *Request*.

Representado pelo diagrama de sequência na Figura 16, o protocolo de interação FIPA *Request* permite um agente (o iniciador) solicitar para outro (o participante) a execução de uma ação. O participante processa o pedido e faz uma decisão de aceitar ou recusar o pedido. Se as condições indicam que um acordo pode ocorrer, então o participante comunica um acordo (*AGREE*), caso contrário uma notificação de declínio é enviada (*REFUSE*).

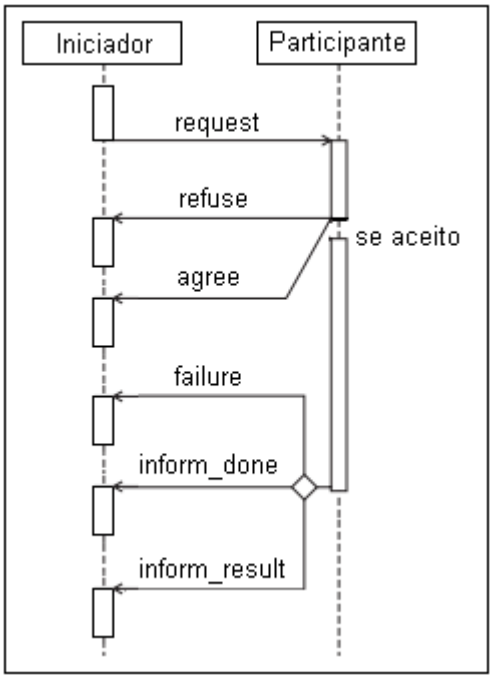


Figura 16: Protocolo FIPA *Request* (BELLIFEMINE, 2004).

Uma vez que o participante aceitou a requisição, processa a ação solicitada e envia uma resposta final, que pode resultar em três situações: se uma falha ocorrer ao realizar a ação

uma mensagem *FAILURE* será enviada, se tudo ocorrer bem uma mensagem *INFORM_DONE* é enviada para indicar tão somente que a ação foi realizada, e há o caso de que não só a informação de ação realizada é enviada, mas também o resultado desta ação, enviando uma mensagem *INFORM_RESULT*.

3.3.3.2 Protocolo FIPA *Contract Net*.

Como um exemplo de um protocolo de interação mais complexa, o protocolo FIPA *Contract Net* descreve o caso de um agente (o Iniciador) que deseja ter alguma tarefa realizada por um ou mais agentes (os participantes) e otimizar uma função que caracteriza a tarefa. Esta característica é geralmente expressa como um custo, mas também poderia ser o tempo mais rápido para conclusão, a distribuição das tarefas, a ordem de retorno da resposta, entre outras. Para uma dada tarefa, qualquer número de participantes pode responder com uma proposta.

O iniciador solicita propostas a outros agentes através da emissão de um convite à apresentação de propostas (*Call for Propose - CFP*), que especifica a tarefa e as condições dos locais sobre a execução desta.

Os participantes que recebem o convite à apresentação de propostas são vistos como potenciais candidatos a serem contratados e são capazes de gerar respostas. As propostas dos participantes incluem as pré-condições do participante, que pode ser o preço, o tempo de execução, o tempo até o início da realização da tarefa, quantidades de tarefas realizadas concomitantemente, entre outras. Os participantes podem se recusar a propor.

As negociações continuam com os participantes que apresentaram proposta para realização (*PROPOSE*), os que enviaram *REFUSE* deixam a negociação. Uma vez que o prazo cessa, o iniciador avalia as propostas e seleciona os agentes para executar a tarefa, com

base em seus critérios previamente definidos. Um, vários ou nenhum agente podem ser escolhidos. Aos agentes da proposta selecionada será enviado um aceite à proposta (*ACCEPT_PROPOSAL*) e os agentes restantes receberão um rejeite à proposta (*REJECT_PROPOSAL*). As propostas são obrigatórias para o participante, de modo que uma vez que o Iniciador aceita a proposta do participante, adquire o compromisso de executar a tarefa. Uma vez que o participante tenha concluído a tarefa, ele envia uma mensagem de conclusão para o iniciador, na forma de uma mensagem de tarefa realizada (*INFORM_DONE*) ou uma versão mais explicativa, na forma de uma mensagem de tarefa realizada com os devidos resultados (*INFORM_RESULT*). No entanto, se o participante não concluir a tarefa, uma mensagem de falha é enviada (*FAILURE*).

A Figura 17 apresenta o diagrama de um protocolo FIPA *Contract Net* com dois participantes, na sua forma geral.

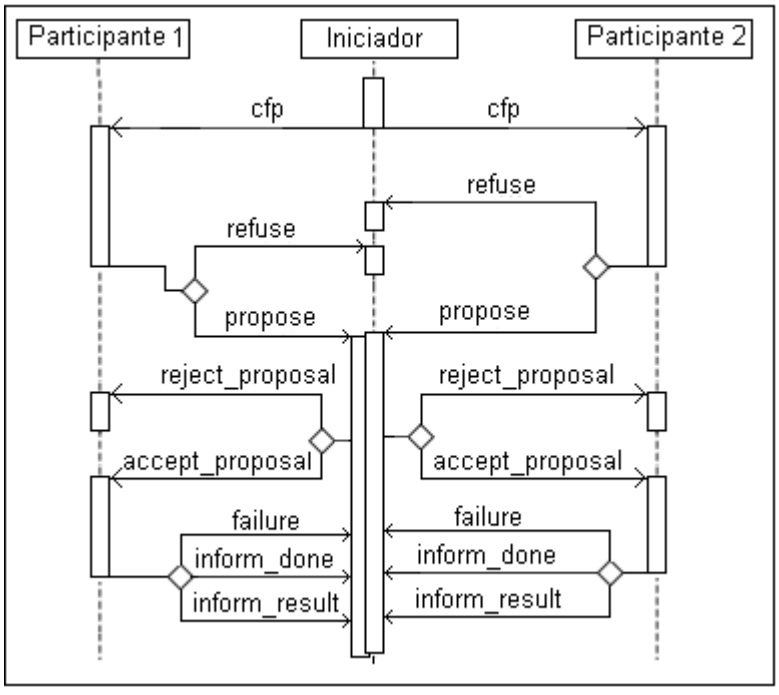


Figura 17: Protocolo FIPA Contract Net (BELLIFEMINE, 2004).

Este protocolo requer que o iniciador saiba quando ele recebeu todas as respostas. No caso de um participante não responder ou propor uma mensagem errada, o iniciador pode ficar esperando indefinidamente. Para se proteger contra isso, o protocolo FIPA *Contract Net* inclui um prazo para que as respostas sejam recebidas pelo iniciador. As propostas recebidas após o prazo serão automaticamente rejeitadas. O prazo é especificado pelo parâmetro de resposta no objeto *ACLMensagem*.

### 3.3.4 Serviço de páginas amarelas em JADE.

Durante a criação de um agente (instância) no container principal, mais dois agentes são criados pelo JADE, que são o *Agent Management System* - AMS e o *Directory Facilitator* – DF. A Figura 18 apresenta esta estrutura.

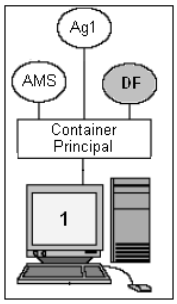


Figura 18: Directory Facilitator (DF) em JADE (BELLIFEMINE, 2004).

O *Directory Facilitator* oferece o serviço de páginas amarelas na plataforma, onde os demais agentes postam suas habilidades e onde os agentes podem buscar o endereço de agentes que possuem habilidades desejadas.

O serviço de páginas amarelas permite aos agentes publicar descrições de um ou mais serviços que prestam, a fim de que outros agentes podem facilmente descobrir e explorá-los. Isto é representado na Figura 19. Qualquer agente pode tanto registrar (publicar) serviços

quanto procurar (descobrir) serviços. As inscrições, modificações, exclusão de registros e pesquisas pode ser realizadas em qualquer momento durante a vida de um agente.

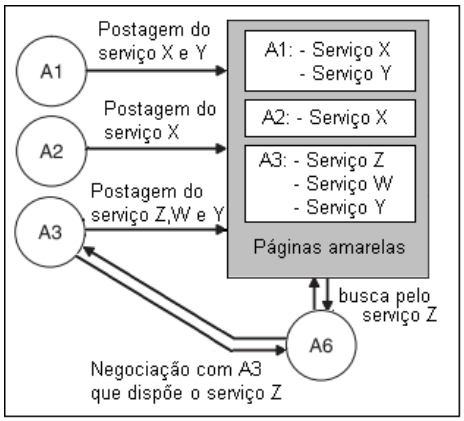


Figura 19: Estrutura de postagem no DF.

Um agente que pretenda publicar um ou mais serviços deve fornecer ao DF uma descrição, que inclui uma lista de serviços prestados e, opcionalmente, a lista de idiomas e ontologias que outros agentes devem usar para interagir com ele. As classes *DFAgentDescription*, *ServiceDescription* e *Property*, incluídas no *jade.domain.FIPAAgent*, tem a função de gerenciar os pacotes e representar as abstrações.

Para publicar um serviço o agente deve criar uma descrição e chamar o método *register*(), método estático da classe *DFService*. A Figura 20 apresenta um código exemplo da operação de registro de serviço no DF.

```
DFAgentDescription dfd = new DFAgentDescription( );
dfd.setName(getAID());
ServiceDescription sd1 = new ServiceDescription( );
sd1.setType("montagem");
sd1.setName("montar azul");
dfd.addServices(sd1);
try{DFService.register(this,dfd);
      } catch (FIPAException fe) {
            fe.printStackTrace( );
       }
```

Figura 20: Código exemplo para postagem no DF.

A busca de serviços ocorre de forma semelhante, onde as características do serviço são preenchidas na instância da classe *ServiceDescription*, e o DF retorna um objeto *Iterator* com a lista de agentes que são capazes de realizar o serviço solicitado. A Figura 21 apresenta um código exemplo para busca de um serviço no DF.

```
DFAgentDescription template = new DFAgentDescription( );
ServiceDescription sd = new ServiceDescription( );
sd.setType("montagem");
template.addService(sd);
try{DFAgentDescription[ ] result = DFService.Search(this, template);
    for( int i = 0; i < reult.length ; i++ ) {
       String out = result[ i ].getName( ).getLocalName( ) + " Faz ";
       Iterator iter = result[ i ].getAllServices( );
       while( iter.hasNext( )) {
           ServiceDescription SD = (ServiceDescription) iter.next( );
           out += "   " + SD.getName( );
       }
    System.out.println(out);
    }
} catch (FIPAException e) {e.printStackTrace( );}
```

Figura 21: Código exemplo para busca de serviços no DF.

# 4. TRABALHOS RELACIONADOS

A recorrência à arquitetura orientada a serviços e sistemas multi-agentes em aplicações computacionais, informática propriamente dita, é algo consensuado, com diversos estudos e aplicações, sendo os resultados aplicados nos sistemas computacionais. Mas a aplicação desses dois conceitos em automação industrial é algo que possui muitas pesquisas, mas poucas aplicações no meio de produção. Dentre as pesquisas estudadas se destacam 6 estudos de casos que apresentam aplicação prática dos paradigmas em sistemas de automação.

## 4.1 SOA EM SISTEMAS DE PRODUÇÃO RECONFIGURÁVEIS

Este é um estudo realizado por Armando W. Colombo, J. Marco Mendes, Paulo Leitão e Francisco Restivo, publicado na Conferência Internacional sobre Informática Industrial – IEEE, Korea, 2008, sob o título "*Service-Oriented Control Architecture for Reconfigurable Production Systems*" (COLOMBO et al., 2008). Neste trabalho os autores abordam o desafio mediante uma proposta de arquitetura de controle modular para sistemas de produção orientada a serviços, para atender aos requisitos fundamentais de flexibilidade e reconfiguração. A arquitetura de controle é baseada em princípios de serviços orientados para alcançar distribuição, modularização, agilidade e sistemas interoperáveis. Complementado com redes de Petri de alto nível (*High Level Petri Network* - HLPN) para o controle do processo e alguns conceitos de multi-agentes e sistemas de decisão para alcançar a autonomia e inteligência para apoiar a resolução de conflitos. A Figura 22 apresenta a proposta do estudo.

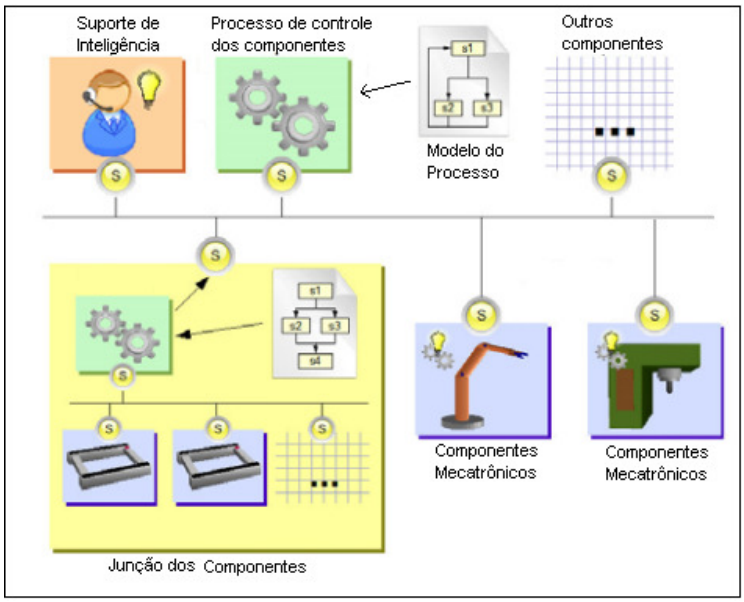


Figura 22: Componentes em um SOA (COLOMBO et al., 2008).

A arquitetura identifica três tipos de componentes que participam do controle: componentes mecatrônicos (*Mecatronics Componets* - MeC), componentes de controle de processo (*Process Control Componets* - PCC) e componentes de inteligência e apoio (*Inteligence Suport Componets* - ISC). Eles controlam seu próprio comportamento e podem ser facilmente combinados para formar dispositivos mais complexos, a fim de construir o sistema de produção desejada, prevendo diferentes níveis de granularidade. O controle local exige, por meio de dispositivos mecatrônicos, a colaboração entre os componentes e controle agregados, para criar serviços de alto nível com base em diferentes quesitos. A reconfiguração automática surge pela fácil reorganização dos componentes e dos serviços que prestam, refletida pela alteração das conexões entre os dispositivos previstos no sistema.

O controle geral é estruturado em módulos adaptáveis e reutilizáveis, incluídos no componente de controle, de acordo com suas necessidades e, eventualmente, implementando uso de tecnologias diferentes. Os módulos propostos não são estritamente necessários para implementar os componentes, portanto, cabe ao desenvolvedor configurar a estrutura de acordo com os objetivos. Os módulos disponíveis são:

a) Módulo de controle lógico: regula o comportamento do componente (o dispositivo físico se incorporado em um MeC, ou um sistema maior se incluído em um componente PCC), através da coordenação de um conjunto de serviços descritos por um modelo lógico. A lógica do módulo de controle pode ser integrada em mais de um componente. A construção de uma arquitetura não hierárquica significa que a supervisão é realmente distribuída;

b) Módulo de comunicação: a comunicação entre os componentes de controle é realizada através da invocação de serviços de componente hospedado e fornecido pelo módulo de comunicação. Como exemplo, um transportador poderá prestar o serviço de transferência para o movimento de paletes. Este serviço pode ser usado por outros componentes, mas também pode fornecer serviços externos, quando necessário (por exemplo, para ser conectado a outro transportador que necessita para solicitar o serviço de transferência de outra transportadora). Uma solução de tecnologias adequadas para implementar o módulo de comunicação orientada a serviços é usar a tecnologia da WEB;

c) Módulo de decisão e módulo manipulador de exceção: a coordenação da execução dos serviços, dependendo da flexibilidade que o sistema revela, requer a tomada de decisões e a resolução de conflitos em tempo de execução, porque um modelo de lógica do sistema não descreve uma sequência fixa de ações, mas sim todas as combinações possíveis dos mesmos. A existência de conflitos não significa rigorosamente que há problemas de projeto, mas deve ser entendida também como uma oportunidade de aplicar a decisão para um sistema mais flexível;

d) Módulo de interface: fornece os mecanismos para integrar os dispositivos físicos, como robôs ou sensores. Como os controladores de dispositivo local normalmente têm arquiteturas fechadas é necessário desenvolver as funcionalidades dos dispositivos físicos. O módulo de interface de dispositivo visa cumprir esse objetivo, fornecendo mecanismos

de apoio à integração fácil e transparente dos dispositivos físicos, dentro de aplicações de controle;

e) Módulo agendador roteador de eventos: é o módulo que fornece as características de mecanismos internos baseado em eventos para passar de um módulo a outro, *buffers* que priorizam eventos em caso de alto tráfego, instalações de proteção e gerenciamento de *thread* e dados, interface geral de *plug-in* de diferentes módulos.

O presente trabalho propôs sistemas de produção reconfiguráveis baseados no conceito de componentes de controle distribuído, que combinam as características de arquiteturas orientadas a serviços, direcionadas para sistemas de automação e produção. Como metodologia de controle de processo, HLPN foi introduzido para criar, validar, simular e executar processos.

## 4.2 SOA UTILIZADO EM SISTEMAS DE DIAGNÓSTICO

Este é um estudo realizado por José Barata, Luis Ribeiro e Armando Colombo, publicado na Conferência Internacional sobre Informática Industrial – IEEE, INDIN 2008, sob o título "*Diagnosis Using Service Oriented Architectures (SOA)*" (BARATA, 2007). Neste trabalho os autores abordam o diagnóstico dos componentes de um sistema orientado a serviços, implementado em laboratório de manipuladores robóticos, onde os agentes monitoram e controlam seus elementos físico-mecânicos.

A proposta de diagnóstico basea-se em módulos dentro dos equipamentos de manipulação, que inspecionam cada parte deste e fornecem dados referentes ao seu estado. O "agente de diagnóstico" pode ter suficiente inteligência programada para avaliar os componentes, suas falhas ou tendências a elas e solicitar, dentro do SOA, um serviço de troca

do componente ou manufatura dele, cabendo aqui à intervenção humana como um agente do sistema. A Figura 23 apresenta esta estrutura.

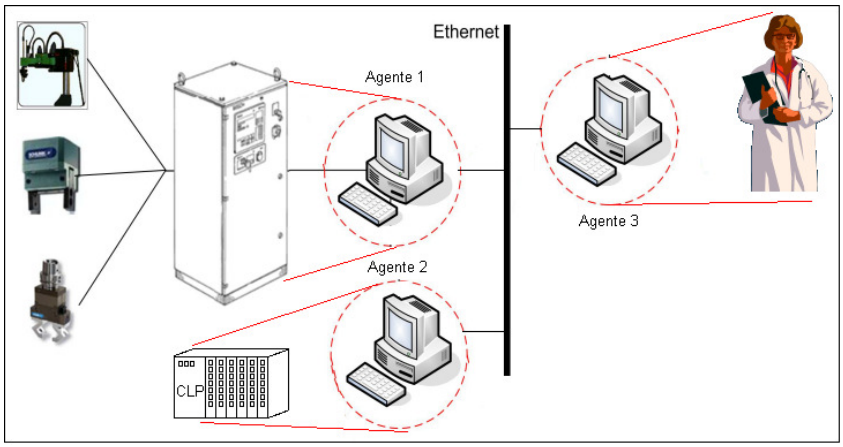


Figura 23: Estrutura de SOA aplicado em diagnóstico de sistemas de manufatura (BARATA, 2007).

As abordagens clássicas para o diagnóstico são baseadas em um "cérebro" centralizado, com alto poder de processamento, que reúne todas as informações sobre o sistema e seu estado. A manutenção preventiva é aplicada e se concentra em avaliar o estado de componentes em uma forma baseada no tempo. Embora essa abordagem possa evitar situações de catástrofe, os custos de manutenção ainda são elevados, devido à periodicidade das inspeções.

É importante notar que, do ponto de vista de SOA, o diagnóstico destas entidades comporta-se como qualquer outra entidade. No nível mais baixo, cada dispositivo "inteligente" controla a seus sensores e avalia seu estado, enquanto em um nível superior as interações dentro de um conjunto de dispositivos de colaboração podem ser monitoradas por outra entidade.

Dada à natureza dinâmica da aplicação SOA, raciocínio probabilístico sobre o sistema não parece viável. É impraticável para avaliar todas as possíveis interações estocasticamente.

Uma maneira natural de diagnosticar é realizando a segmentação nestes pequenos dispositivos, usando o conhecimento de falhas relacionadas.

No caso de estudo, as garras dos robôs existentes estão equipadas com sensores que detectam a sua abertura e fechamento. No caso da pinça, as leituras dos sensores indicando deteriorização podem não ser suficiente para desencadear uma ação de autocura, mas um operador humano pode interagir com o sistema e evitar a falha. O mesmo raciocínio se aplica ao resto do sistema.

## 4.3 SOA EM UMA PLATAFORMA PARA CÉLULAS ROBÓTICAS INDUSTRIAIS

Este é um estudo realizado por Nilsson, Pires e Veiga, publicado no *SMErobot*™ - *The European Robot Initiative for Strengthening the Competitiveness of SMEs in Manufacturing*, sob o título "*On the use of Service Oriented Software Platforms for industrial robotic cells*" (NILSSON; PIRES; VEIGA, 2007). Neste trabalho os autores abordam uma real implementação de SOA baseada em uma plataforma *middleware*. Essa plataforma inclui mecanismos adequados de apoio ao SOA, com as características principais de: endereçamento, descoberta, controle, descrição e eventos. O estudo aplica-se em células robóticas industriais.

A proposta é o desenvolvimento de uma rede de dispositivos chamados *universal plug and play* (UpnP). Como os dispositivos estão em plataformas diferentes, o SOA, por sua fácil adaptação a sistemas com plataformas distintas, vem atender a esta necessidade de conexão de diferentes sistemas, com a funcionalidade de dispositivos UPnP.

Este trabalho aponta os efeitos do uso de arquitetura orientada a serviços em automação, num comparativo entre os dias atuais e o futuro com a utilização de SOA, sendo descritos na tabela 2.

Tabela 2: Aspectos do SOA aplicados em Automação.

| Aspecto | Dias atuais | Futuro |
|---|---|---|
| Sistema | Grande centralização e controladores inteligentes. | Inteligência descentralizada e componentes autônomos. |
| Comunicação | Conexão cliente servidor, ponto a ponto. | Publicação e consulta, pares a pares. |
| Pré-ajuste | Longo e difícil, programação manual. | Não há programação, *plug and play*, de fácil configuração. |

## 4.4 MAS EM SISTEMAS DE MANUFATURA INTEGRADOS E DISTRIBUÍDOS.

Este é um estudo realizado por José Barata, Raymond Boissier, Camarinha-Matos, Mohammed Raddadi e Francisco Restivo, sob o título "*Integrated And Distributed Manufacturing, A Multi-Agent Perspective*" (BARATA et al., 2001). Neste trabalho os autores propõem o desenvolvimento de aplicações de manufatura distribuída com o uso de sistemas multi-agentes. Os agentes são definidos no trabalho como um componente de software e / ou *hardware*, capaz de agir e de tomar decisão, a fim de realizar tarefas.

Seja qual for a abordagem, hierárquica ou holônico, o funcionamento eficiente das aplicações de manufatura distribuída se baseiam na comunicação entre os diferentes processos industriais ou equipamentos de monitoramento. Como forma de esforço de programação há uma proposta da recorrência a ambientes multi-agentes como AgentBuilder, Concórdia, JATLite, IBMAglets, FIPA-OS e JADE. Dentre análise, os autores escolheram a implementação com JADE, utilizando FIPA-ACL, que descreve todo ato comunicativo como uma forma narrativa, formal e semântica, baseada na lógica modal, e também inclui uma descrição normativa de um conjunto de protocolos de alto nível de interação como uma ação solicitada.

Os agentes que representam os dispositivos físicos, tais como sensores, robôs, máquinas CNC, etc., devem ter um módulo de interface que implementa a comunicação entre a parte lógica de um agente e um dispositivo físico. Os autores apontam este processo como "agentificação", ou seja, tornar um componente físico um agente. A Figura 24 apresenta o esquema deste processo.

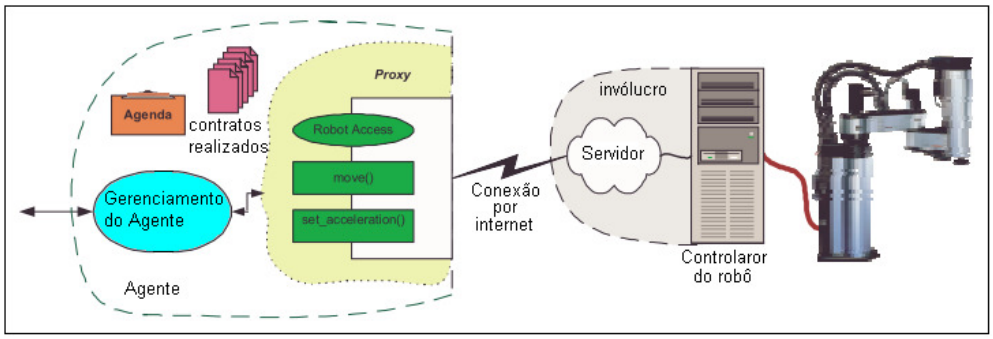


Figura 24: Integração dos componentes físicos no MAS (BARATA et al., 2001).

O agente não precisa necessariamente estar junto ao componente físico, mas conectado a ele. Na proposta de trabalho os autores mostram o gerenciamento do agente se conectando ao controlador do robô através de um *proxy* e servidor, através da internet. O processo de "agentificação" ocorre de forma remota.

## 4.5 MAS APLICADO EM SISTEMAS DE TRANSPORTES.

Este é um estudo realizado por Pavel Vrba e Vladimír Maríık, publicado no *IEEE Transactions on Systems, Man, And Cybernetics*, sobre o título "*Capabilities of Dynamic Reconfiguration of Multiagent-Based Industrial Control Systems"* (VRBA; MARIK, 2010). Este estudo propõe o uso de sistemas multi-agentes em sistemas de transportes de materiais, onde agentes se integram modularmente e compõe rotas, sendo que cada rota possui um custo

atribuído. O sistema de controle se vale das funcionalidades dos agentes para determinar a rota mais econômica. Os agentes desta rota se agrupam para formar este caminho e executar o translado.

A tarefa básica de qualquer sistema de controle discreto de fluxo de materiais é o transporte de entidades como: matérias primas, peças, conjuntos, pacotes, etc., entre os locais no processo de fabricação. O sistema de controle de fluxo deve ser capaz de desviar peças de trabalho em uma rede complexa e redundante de rotas de transporte. No transporte, rotas são geralmente implementados como pistas de transporte com desvios. Diferentes critérios de otimização são aplicados, geralmente o caminho mais curto, tempo percorrido, maior velocidade alcançada, equilíbrio de carga, entre outras.

Como sistema de transporte, o trabalho usa uma proposta da Rockwell Automation, uma simulação baseada em agentes reais com um controle de tempo de execução exclusivo, baseado em interfaces com CLP. Esta relação permite aos agentes interagir com qualquer subsistema de simulação ou sistema físico, compartilhando os valores dos sensores e atuadores na memória do CLP. A parte principal deste sistema MAST é uma biblioteca de agentes que atualmente contém 16 classes que representam diferentes componentes de fabricação, como desvio da célula de trabalho, transportador, sensor, identificação por leitor de radiofrequência (RFID), robô, etc. Do ponto de vista da tarefa de transporte, as células de trabalho são consideradas como locais no chão da fábrica ligadas através de uma rede de transportadores, entre as quais as peças de trabalho são transportadas. Sugere-se que a célula de trabalho tem um ou mais transportadores de entrada através da qual os produtos entram, e um transportador de saída é usado para enviar os produtos para fora do sistema. A Figura 25 apresenta uma imagem e uma simulação deste sistema descrito.

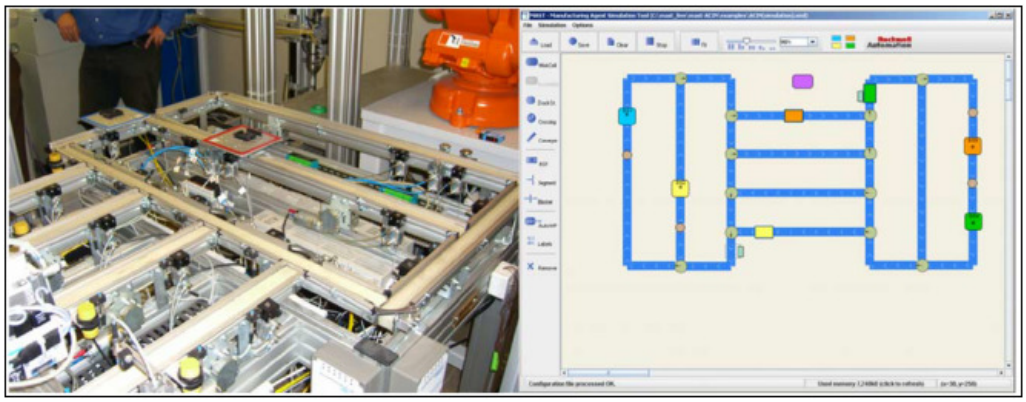

Figura 25: Sistemas de transporte utilizando MAS (VRBA, 2010).

As negociações entre os agentes ocorrem de forma a obter o caminho mais curto para o transporte entre as células de trabalho. Cada caminho é representado por um agente e a agregação de agentes configura caminhos diferentes para alcançar o mesmo ponto final do transporte. Então, um agente pode ser retirado do sistema sem que haja distúrbio na produção, pois as negociações ocorrem dinamicamente e a reconfiguração ocorre na próxima negociação, assim como a inserção de agentes de transportes, que passam a concorrer na negociação assim que são ligados ao sistema.

## 4.6 MAS APLICADO EM SISTEMAS REMOTOS DE MANUTENÇÃO.

Estudo realizado por Yu Ren, Ye Luqing e Fu Chuang, publicado na Intelligent Agent Technology, IAT 2004, sob o título "*A Multi-Agent-Based Remote Maintenance Support and Management System*" (REN, 2004). O objetivo do sistema desenvolvido pelos autores foi integrar as atividades relacionadas à manutenção de uma empresa industrial e utilizar todos os recursos disponíveis para certificar-se que a manutenção está no tempo certo, local certo, e com o método certo, com o uso de estrutura multi-agente. A Figura 26 apresenta a estrutura proposta.

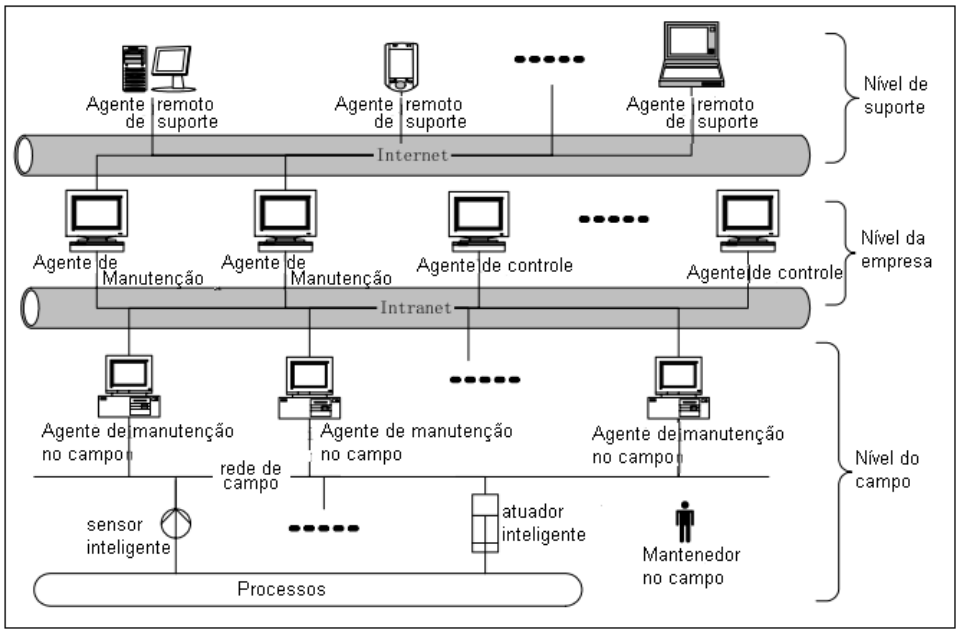


Figura 26: Sistema de manutenção implementado com MAS (REN, 2004).

O sistema pode ser dividido em três níveis. O mais baixo é o nível de campo. Aqui os processos industriais são supervisionados e controlados pelos sensores e atuadores inteligentes. Para cada problema de manutenção, há uma lista de agentes especialistas que se relacionam com este problema e chamam a lista de agentes especialistas em manutenção. O agente de controle vai reunir todas as informações dos dispositivos de campo e entregar as informações necessárias para cada agente especialista. Todos os agentes especialistas que se inscrevam para a discussão do problema formam a lista dinâmica de agentes especialistas. Começam então as discussões dos problemas, trocam opiniões, lançam explicações sobre sua sugestão, de forma a chegar a um consenso. Este consenso será a solução mais completa possível.

Todos os agentes que atuam no sistema são do tipo agentes de colaboração, inteligentes e reativos, que enfatizam a cooperação com outros agentes.

## 4.7 RESUMO DOS TRABALHOS RELACIONADOS

SOA e MAS não são paradigmas que se contrapõem, e em muitas aplicações acabam por se complementarem. A tabela 3 compara os trabalhos relacionados descritos nas seções anteriores.

Tabela 3: Correlação entre os estudos de casos.

| Caso (seção) | Paradigma | Aplicação | Meio de comunicação | Atuação dos dispositivos |
|---|---|---|---|---|
| 4.1 | SOA. | Sistema de automação. | Serviços WEB (TCP/IP). | Como elementos que dispõem serviços, onde um agente é o escalonador de tarefas, |
| 4.2 | SOA. | Diagnóstico em sistemas de automação. | Serviços WEB (TCP/IP). | Como elementos inspecionadores de equipamentos, fornecendo como serviço os dados relativos ao estado do equipamento, também negociando o momento de realizar inspeção e intervenção. |
| 4.3 | SOA. | Implementação de dispositivos *plug and play*. | Serviços WEB (TCP/IP). | Os dispositivos são modelados para conexão e execução assim que plugados no sistema, dispondo e solicitando serviços |
| 4.4 | MAS. | "Agentificação" de componentes físicos. | TCP/IP, UDP. | Os agentes que implementam as funcionalidades, dispondo e solicitando tarefas. São criados de forma remota ao componente físico. |
| 4.5 | MAS. | Sistemas de transportes. | Sobre CLP. | Os agentes disponibilizam funcionalidade de transporte de materiais, sendo a composição entre os agentes que formam o caminho de transporte. |
| 4.6 | MAS. | Sistema de suporte a manutenção. | Serviços WEB (TCP/IP). | Agentes recebem dados dos elementos físicos e se relacionam para buscar a melhor solução para um problema que se apresenta. |

Nota-se aqui que as aplicações podem ser distintas, mas o conceito de SOA e MAS continuam os mesmos, pois está se falando de dispositivos que postam seus serviços e interagem uns com os outros para formar uma cadeia social de cooperação entre agentes.

Os trabalhos abordam a aplicação de conceitos SOA e MAS em sistemas de manufatura, mas não comparam com outros sistemas de gerenciamentos, como é o caso dos controladores lógicos programáveis, onde seria significativo observar em que tópicos o conceito SOA e MAS têm vantagens ou desvantagens em relação aos sistemas de gerenciamento tradicionais. O trabalho proposto nesta dissertação se dispõe, quando da validação, que as métricas sejam comparadas com a mesma manufatura, mas implementada com gerenciamento por CLP, segundo norma IEC61131. Com isto pode-se definir que em dada aplicação um ou outro conceito seja mais vantajoso.

## 5. PROPOSTA DO TRABALHO.

As abordagens tradicionais tendem a visualizar o sistema de manufatura numa visão *top-down* que considera inicialmente o sistema como um todo e partir então para uma especificação genérica dos módulos que o compõem. Subseqüentemente, a descrição desses módulos é refinada, possivelmente recorrendo à partição deles em sub-módulos, os quais são então detalhados, até atingir os elementos básicos do sistema. O sistema de gerenciamento da manufatura é encarregado pela manufatura dos produtos, valendo-se dos sistemas de produção como recursos ao seu gerenciamento.

A proposta do trabalho é uma visão bottom-up, que considera inicialmente a descrição detalhada dos elementos básicos que o compõem. Esses elementos básicos são então agregados, possivelmente em vários níveis, até que uma descrição completa do sistema seja obtida. Os recursos ficam a disposição da manufatura, ofertando serviços e dispondo-se a coligações.

E nesta visão os elementos são modulares. Agentes com funcionalidades autônomas, que podem se interfacear por software e hardware com os demais agentes do sistema, somam funcionalidades de forma que o conjunto de módulos forme um serviço desejado. O sistema de manufatura passa a dispor características de modularização e reconfiguração.

A disposição dos serviços segue o conceito de Arquitetura Orientada a Serviços – SOA, mas a proposta do trabalho utilizará algumas das características do SOA pertinentes a sistemas de manufatura industrial. Recursos como *WEB-services* e *Device Profile for WEB Services – DPWS* não são utilizados.

É sabido que entre os conceitos MAS e SOA há uma região em que características dos sistemas são comuns (BATARA, 2008), o que possibilita os conceitos serem complementares, Como proposta de abordagem de uma manufatura utilisando os conceitos SOA e MAS, a manufatura vale-se das características comuns e complementares de SOA e MAS.

E com referência aos estudos de caso que foram realizados no Capítulo 4, os tópicos que foram relacionados para formar a proposta conceitual deste trabalho são:

a) Na secção 4.1, SOA aplicado em sistemas de automação, foi tido como referência a modularização dos elementos do sistema de manufatura, na sua disposição de serviço;
b) Na secção 4.2, SOA utilizado em diagnósticos de sistemas de automação, a aplicação de um agente sobre um equipamento de manufatura já concebido, no processo de "agentificação";
c) Na secção 4.3, SOA em plataformas para células robóticas industriais, a funcionalidade dos dispositivos tidos como Universal Plug and Play remetem a aos elementos da manufatura serem concebidos de forma a serm inseridos ou retirados do sistema, sem que este necessita parar seu processo ou ser reprogramado;
d) Na secção 4.4, MAS em sistemas de manufatura integrados e distribuídos, traz a abordagem de que o agente não necessita estar fisicamente acoplado ao dispositivo de hardware, mas remotamente conectado. Esta característica é importante para agentes móveis, como é o caso de um palete que deve se movimentara ao longo da esteira;
e) Na secção 4.5, MAS aplicado em sistemas de transportes, aborda as coligações realizadas entre agentes para formar um serviço desejado, o que ocorrer pode ocorrer neste trabalho no momento em que duas ou mais funcionalidades de recursos distintos necessirarem serem coligadas para formar um serviço desejado;
f) Na secção 4.6, MAS aplicados a sistemas remotos de manutenção, traz como contribuição a próatividade dos agentes em coligarem-se para formar um concenso , que se traduz no serviço prestado. É utilizado como referência de conceito, mas sua aplicação direta não ocorrerá ao longo da proposta conceitual.

## 5.1 PROPOSTA CONCEITUAL

O desenvolvimento de sistemas de manufatura na abordagem tradicional remete a identificação dos componentes do sistema segundo a sequência operacional de produção. A partir desta sequência descrevem-se os elementos de manufatura como partes sequenciais de um sistema de produção maior e programa-se um controlador central para que a sequência de manufatura ocorra. Algumas abordagens mais modernas consideram o projeto e controle de sistemas de manufatura de forma a identificar na sequência operacional de produção atividades semelhantes, padronizando tarefas semelhantes e identificando tempos de processos que onerem maior disponibilidade de equipamentos. A essência é decompor os sistemas de manufatura em vários subsistemas, ou grupos, controláveis e dispostos na forma de manufatura celular, em que partes similares são agrupadas em famílias e máquinas são agrupadas em células. A abordagem por células de manufatura fornece mais diversidade e flexibilidade de produção, mas ainda deixam a desejar quando lhes é solicitado a reconfiguração rápida e eficiente.

Este trabalho propõe que um sistema de manufatura seja abordado com suas etapas dissociadas, funcionalmente independentes e correlacionadas para formar um sistema de manufatura maior. A Figura 27 apresenta um exemplo destas abordagens.

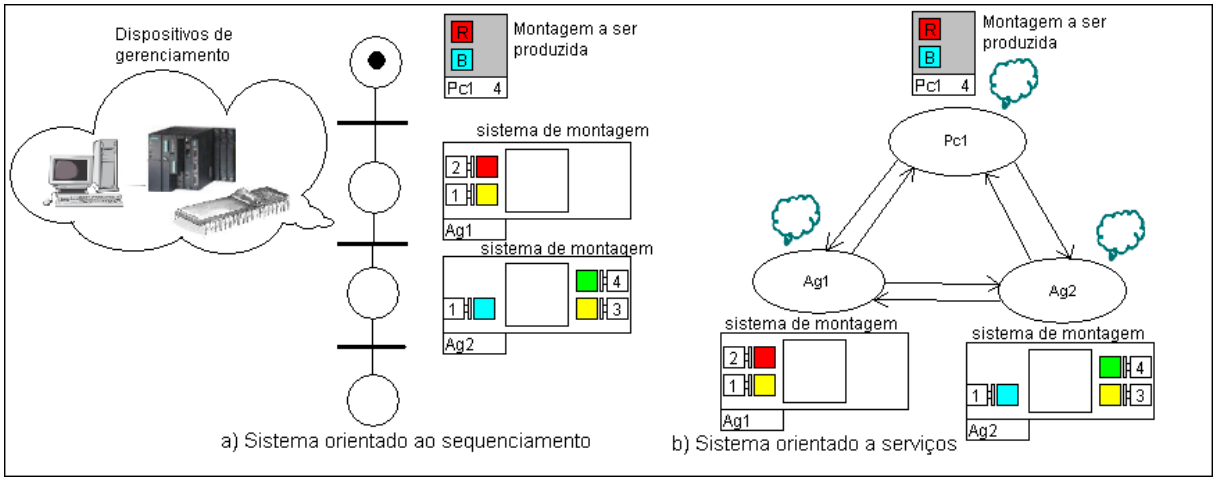


Figura 27: Sistema de manufatura na abordagem sequencial e abordagem orientada em SOA/MAS.

No sistema orientado sequencialmente o produto requisitado, montagem de peça azul e vermelha, deverá passar primeiro pela estação Ag1 para montar a peça vermelha e, na sequência, deverá passar pela estação Ag2 para montar a peça azul. Já na abordagem orientada a serviços, o mesmo produto solicitado terá a possibilidade de passar pela estação Ag1 ou pela estação Ag2, sendo a sequência de montagem definida em uma negociação entre os elementos do sistema. A visão sequencial deixa de ser o foco e a relação entre os processos de manufatura através de negociação e coalizão passa a ser a meta.

Para o desenvolvimento de sistemas de manufatura com abordagem em componentes independentes, que se relacionam e interagem para executar tarefas descritas no próprio produto são usados os conceitos de arquitetura orientada a serviços e sistemas multi-agentes. A abordagem do processo de produção neste conceito pode ser especificado em 5 etapas, como descrito a seguir:

1ª Etapa: identificação dos componentes que tem capacidade de funcionar autonomamente.

Nesta etapa deve-se observar o processo de manufatura requerido e agrupar os componentes segundo processo que realizam. Em seguida é verificado se cada um destes

grupos de componentes tem capacidade de operar autonomamente, como uma célula individual. Neste momento são identificados os agentes de recursos (RA) e os candidatos a agentes de transporte (TUA).

2ª Etapa: verificar se os elementos de transporte devem ser mapeados como agentes de transporte (TUA).

Para que haja a conexão física do produto com os elementos de manufatura (agentes de recursos) deve haver sistemas que transportem o produto em manufatura de um elemento a outro. Estes sistemas são chamados de sistemas de transporte. Os sistemas de transporte podem ser do tipo contínuo, onde o caminho entre os elementos já é determinado e está sempre ativo, bastando o produto ser colocado em uma parte deste sistema de transporte para ser encaminhado até os outros elementos de manufatura. Mas o sistema de transporte pode ser seletivo, onde o produto tem mais de uma possibilidade de itinerário para chegar a diferentes elementos de manufatura. O que definirá se o sistema de transporte será ou não mapeado como agente de transporte (TUA) é sua autonomia. Se ele é capaz de funcionar autonomamente, e tem a capacidade de dispor este serviço de transporte e seu estado funcional aos demais elementos do sistema de manufatura, então ele deve ser mapeado como agente de transporte.

3ª Etapa: considerar cada produto ou conjunto de produtos como um agente de produto (PA).

Este é um tópico diferenciado em relação às demais abordagens, onde o sistema de gerenciamento da manufatura é responsável pela entrada de produto a ser manufaturado e também pela dispensa deste quando acabado. Na abordagem da manufatura em SOA/MAS o produto é um elemento que dispõe e solicita serviços, com capacidade de comunicar-se com os demais elementos do sistema de manufatura e que tem capacidade de processamento para

alocar a sequência operacional de manufatura requerida. Como o produto necessariamente deverá deslocar-se entre os elementos de manufatura, é sensato imaginar este como um elemento móvel que deverá ter capacidade de processamento, para poder comunicar-se e gereciar a alocação de recursos do sistema. Isto pode ser um problema para este tipo de abordagem, em face de baixa disponibilidade de tecnologias que tenham capacidade de processamento e que possam serem embarcados em elementos móveis ou conectados remotamente a dispositivos computacionais fixos que o represente logicamente.

4ª Etapa: promover a implementação dos dispositivos como agentes, implementando em cada agente um módulo de interligação com o hardware, um módulo lógico que gerencie o dispositivo e um módulo de comunicação que possibilite a troca de mensagens com os demais agentes.

Este processo é onde os agentes identificados são concebidos em *hardware* e *software*. Basicamente são implementados 3 módulos: módulo de comunicação, módulo lógico e módulo de hardware.

Como pode ser observado na Figura 28, o módulo de comunicação é o responsável por prover a troca de mensagens deste agente com os demais agentes do sistema, a fim de que serviços sejam disponibilizados ou alocados. Neste módulo também é definido o meio de comunicação em rede.

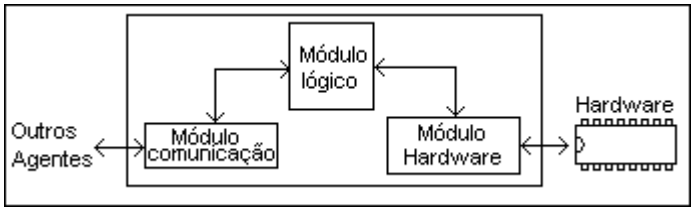


Figura 28: Estrutura de um agente implementado.

O módulo lógico é responsável por executar o agente de forma autônoma. É neste módulo que é definido o comportamento do agente, desde a forma com que irá interpretar as mensagens que chegam a ele, até a forma com que comandará a execução do *hardware*, também definindo as mensagens que serão enviadas por ele a outros agentes. Cada agente implementado, deverá executar em seu processo de inicialização um programa próprio que define seu comportamento, o que inclui se registrar no agente de identificação, postar serviços no agente de identificação de serviços, se dispor a atender a solicitações de outros agentes ou solicitar a outros agentes a alocação de um serviço específico.

Junto ao processo de promover a implementação dos dispositivos como agentes são concebidos 2 agentes que comporão o sistema de manufatura orientada a serviços. O agente de identificação (AMS) é um dos agentes que devem ser implementados na plataforma que suportará os agentes, e que terá a função de receber a identificação de cada agente que se conectar a esta plataforma. Deste modo, qualquer agente poderá solicitar ao AMS a informação de quais outros agentes compõem o sistema e qual sua identificação na plataforma. Já o agente de identificação de serviços (DF) é responsável por receber a postagem dos serviços disponíveis de cada agente que lhe contatar. Assim, um agente pode contatar o DF e solicitar uma lista de agentes da plataforma que realizam um determinado serviço.

5ª Etapa: interligar os agentes em uma plataforma de comunicação entre agentes.

Esta etapa consiste em prover um meio físico para comunicação em rede e prover um protocolo de comunicação entre cada agente, de forma a possibilitar que todos possam contatar a cada um dos demais, através de seu identificador na plataforma. O protocolo deve conter uma semântica que possibilite uma troca de informações de fácil e de rápida

interpretação, ou seja, mensagem que identifiquem: o remetente, o destinatário, o conteúdo e a atividade a ser executada.

A Figura 29 apresenta um diagrama de sequência de comunicação entre agentes de forma conceitual. Neste diagrama se pode observar que a negociação deve começar pela busca por parte do agente produto de agentes de recurço que possam lhe atender em serviço desejado. Uma vez encontrado o agente recurso, então há uma busca por agentes de transporte que prestem o serviço de transporte do produto até o agente de recurso descoberto. Uma vez identificado o agente de serviço que executa a tafera desejada e o agente que possa promover o transporte, então primeiramente o agente de transporte é alocado e na sequência o agente de recurso é alocado para executar a tarefa.

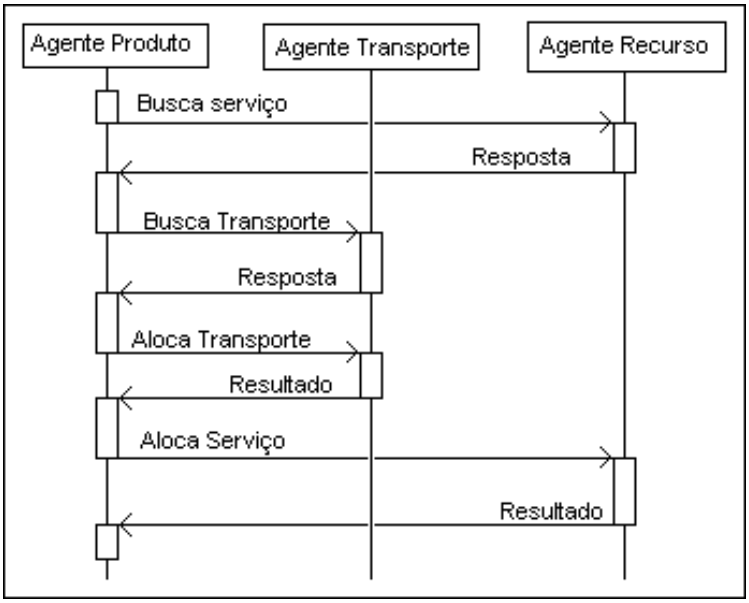


Figura 29: Diagrama de sequência de comunicação entre agentes na forma conceitual.

## 5.2 EXEMPLO DE APLICAÇÃO DA PROPOSTA CONCEITUAL

Para melhor compreender a proposta conceitual descrita na seção anterior, considere o sistema de manufatura apresentado na Figura 30, o qual contém 3 estações de produção distintas e um sistema de transporte que leve o produto a cada estação.

Numa visão tradicional este sistema poderia ser modelado através de uma rede de Petri, conforme a Figura 31 que apresenta o sistema de montagem, inicialmente projetado para ser gerenciado de forma sequencial.

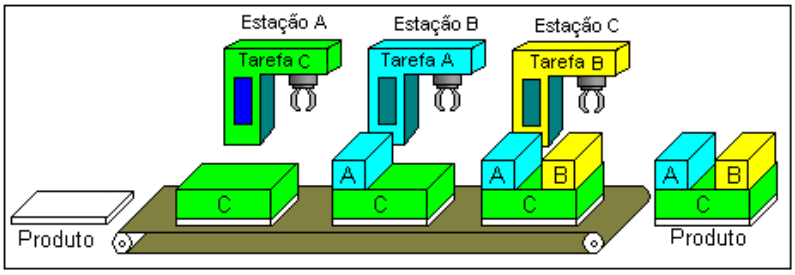


Figura 30: Sistema de montagem gerenciado de forma sequencial.

Neste caso, se um sistema de montagem falhar, o sistema todo fica comprometido, assim como a inserção de mais um sistema de montagem é dificultada por necessitar reprogramar toda a sequência.


Figura 31: Rede de Petri que modela o sistema de montagem.

Conforme já descrito anteriormente, a proposta conceitual propõe a concepção de um sistema de automação da manufatura no conceito de arquitetura orientada a serviços – SOA, onde cada estação de montagem é vista como um sistema autônomo, sendo seus serviços disponibilizados aos demais componentes do sistema. Também se propõe que cada componente tenha proatividade para negociar com outros componentes sua alocação de serviços/tarefas. Para isto, cada componente do sistema é modelado como um agente, numa

plataforma multi-agentes – MAS. Nota-se então que a proposta conceitual de implementação do sistema de automação em uma arquitetura orientada a serviços, requer a utilização das funcionalidades de sistemas multi-agentes. O que se tem é uma fusão de MAS e SOA.

Aplicando a proposta conceitual em suas etapas tem-se:

1ª Etapa: identificação dos componentes que tem capacidade de funcionar autonomamente.

Neste caso identifica-se os seguintes componentes que têm capacidade de funcionar autonomamente:

a) Estação de montagem A;
b) Estação de montagem B;
c) Estação de montagem C;
d) Sistema de transporte por esteira.

2ª Etapa: verificar se os elementos de transporte devem ser mapeados como agentes de transporte (TUA).

Analisando a manufatura com uma visão de *hardware*, observa-se que há um fluxo único de processo, o que leva a propor um sistema de esteira linear para transporte do produto até as estações de manufatura e sua posterior saída de produto acabado. Também deve prover o deslocamento do produto do sistema de transporte para as estações de manufatura e sua posterior volta como produto manufaturado ao sistema de transporte.

Como resultado desta análise, o sistema de transporte tem capacidade de se comportar como um sistema autônomo, onde o serviço prestado é o transporte do produto de uma posição a outra.

3ª Etapa: considerar cada produto ou conjunto de produtos como um agente de produto (PA).

Agora o produto é que deterá informações quanto ao fluxo de montagem, comunicando-se com os demais sistemas e alocando serviços. É comum ver o produto como um palete, com a capacidade de agregar mais de um produto, assim considerado como um único agente que representa um conjunto de produtos similares.

A capacidade de processamento não necessita estar diretamente embarcada no palete, mas pode estar disponível em outro local do sistema, sendo que o palete deve possuir um identificador único que o localize fisicamente dentro do sistema. O sistema remoto que representará o agente produto fará as devidas conexões com os demais agentes em rede.

4ª Etapa: promover a implementação dos dispositivos como agentes

Até aqui já foram identificados seguintes agentes:

a) Agente de recursos RA1: responsável pela manufatura da estação A;
b) Agente de recursos RA2: responsável pela manufatura da estação B;
c) Agente de recursos RA3: responsável pela manufatura da estação C;
d) Agente de transporte TUA1: responsável pelo transporte do produto no sistema de manufatura;
e) Agente de produto PA: responsável por gerenciar o fluxo de processo por onde o produto irá passar, identificando e alocando serviços.

Os agentes devem conter o módulo de *hardware* que fará o gerenciamento dos dispositivos atuadores e sensores, comandando a manufatura propriamente dita. Este módulo pode comandar diretamente os dispositivos ou então comunicar-se com um microcontrolador ou controlador lógico programável, enviando comandos para estes executarem as tarefas e informarem o estado da manufatura.

Também devem conter o módulo lógico onde será programada a estrutura do agente, com o gerenciamento dos serviços disponíveis, sua alocação e sua disponibilidade. O programa de controle deste agente deve incluir código para recebimento de mensagens, sua interpretação e execução da lógica definida em resposta as mensagens definidas. Como primeira tarefa ao ser executado, este agente deve comunicar com o agente de identificação (AMS) para registrar sua presença na plataforma. Caso ele disponha de serviços, então deve contactar o agente de identificação de serviços (DF) para registrar os serviços que estes podem prestar.

E, por fim, o módulo de comunicação deve ser implementado, o qual trocará mensagens com outros agentes da plataforma. Neste módulo serão definidas as camadas de rede que serão utilizadas, incluindo protocolo e meio físico. O objetivo é propiciar uma comunicação que seja transparente para o módulo lógico.

Além dos agentes definidos, também será necessário conceber o agente de identificação (AMS) e o agente de identificação de serviços (DF). O AMS será o primeiro agente a ser executado na plataforma, pois à medida que outros agentes forem executados, estes entrarão em contato, tão logo inicializados, com o AMS para registro de suas identificações. Assim, o AMS conterá a identificação individual de cada agente que estiver na plataforma, dispondo também do serviço de consulta a esta lista. Já o DF, com quem os demais agentes da plataforma manterão contato para divulgar os serviços que disponibilizam, também disponibilizará um serviço de consulta, onde os agentes podem solicitar uma lista de agentes que tem capacidade de realizar uma tarefa específica.

5ª Etapa: interligar os agentes em uma plataforma de comunicação entre agentes.

Esta etapa final propõe a implantação de uma plataforma de comunicação entre os agentes, através de uma rede que interligue os módulos de comunicação de cada agente do

sistema, dispondo um meio físico com domínio único, em que cada agente tenha uma identificação própria.

A Figura 32 apresenta a correlação entre o sistema de automação e seus agentes. Pode-se identificar a relação existente de cada componente com a arquitetura orientada a serviços, utilizando plataforma multi-agentes.

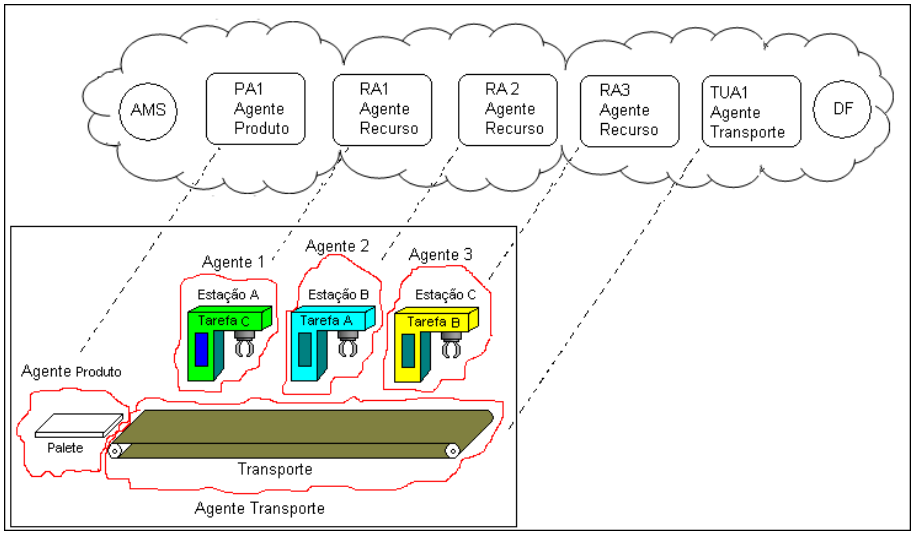


Figura 32: Identificação dos agentes no sistema de automação.

Esta concepção de arquitetura SOA/MAS para automação da manufatura propicia ao sistema a capacidade de inserção de novas estações (agentes) e remoção de estações sem que haja interferência no funcionamento geral, pois a arquitetura proposta suporta a reorganização do processo de manufatura, apresentando flexibilidade para exercer operações diferentes em cada estação e agilidade quanto à produção de produtos diferentes.

# 6. IMPLEMENTAÇÃO

## 6.1 CONCEPÇÃO DE UM SISTEMA DE MANUFATURA FLEXÍVEL PARA USO COMO DEMOSTRADOR

A fim de que a proposta conceitual possa ser validada, foi escolhida uma manufatura industrial para realização de ensaios. Esta manufatura se resume a um produto que é a montagem de 1 a 4 blocos coloridos em posições definidas em um palete. Cada uma destas montagens pode representar um processo de manufatura destro da indústria, como: furação, rosqueamento, montagem, etc. Um palete que possui 4 posições é disposto para acondicionar o produto, em que na posição 1 são montados cubos azuis, na posição 2 são montados cubos vermelhos, na posição 3 são montados cubos amarelos e na posição 4 são montados cubos verdes. As estações de montagem têm capacidade de montar de 1 a 4 cubos distintos, sendo definido quando da sua instalação no sistema. A Figura 33 mostra o croqui da manufatura proposta, o produto a ser manufaturado e o croqui da estação de montagem.

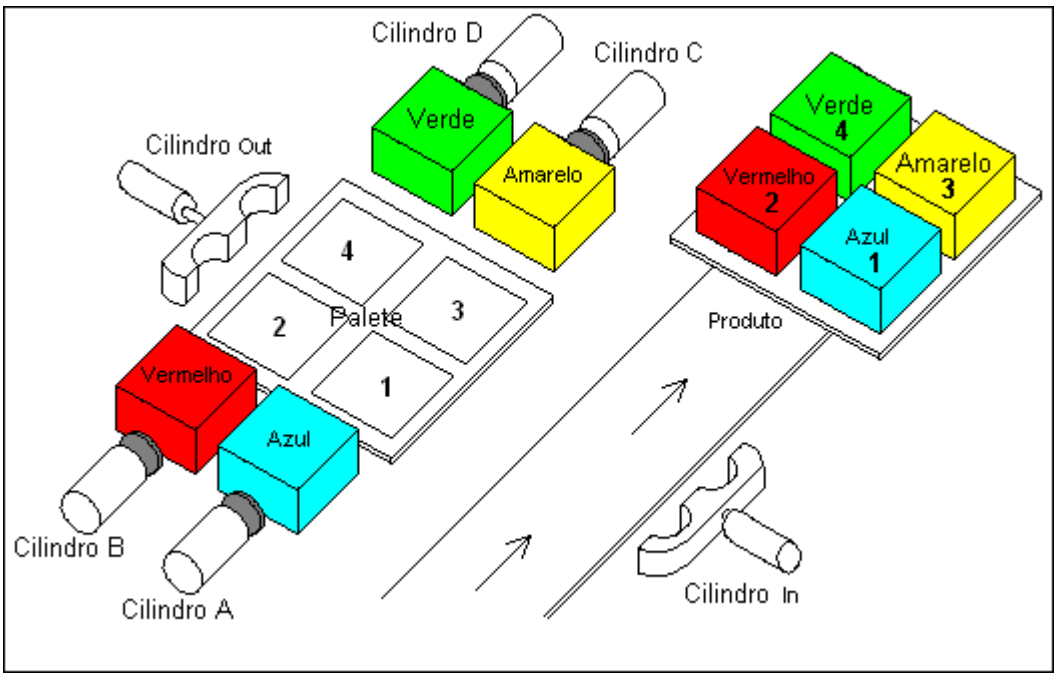


Figura 33: Manufatura proposta para validação.

O meio de transporte do produto entre as estações de montagem é realizada por uma esteira transportadora, formando um ciclo de transporte fechado. Uma vez que o produto entra na esteira ele fica deslocando-se em um circuito fechado, passando à frente das estações de montagem. A entrada e saída de produto na esteira são realizadas de forma manual, pelo operador do sistema.

A estação demonstrada na Figura 33 faz a montagem de peças azuis na posição 1 do palete através do cilindro “A”, peças vermelhas na posição 2 do palete através do cilindro “B”, peças amarelas na posição 3 do palete através do cilindro “C” e peças verdes na posição 4 do palete através do cilindro “D”. O cilindro “In” é responsável pela entrada do palete da esteira transportadora para a estação de montagem, já o cilindro “Out” é responsável pela saída do palete da estação de montagem de volta à esteira. O sistema completo será composto com até 6 estações de montagem de peças, sendo que até 6 paletes de produto poderão estar inseridos no sistema concomitantemente.

Cabe salientar que esta manufatura proposta diz respeito a um processo bastante simples que é a montagem de um cubo, mas que em um processo fabril pode representar uma operação de usinagem, uma montagem de conjunto, um processo de teste de qualidade, entre outros. Então, a manufatura proposta pode perfeitamente representar uma manufatura industrial.

A Figura 34 apresenta a proposta de manufatura completa.

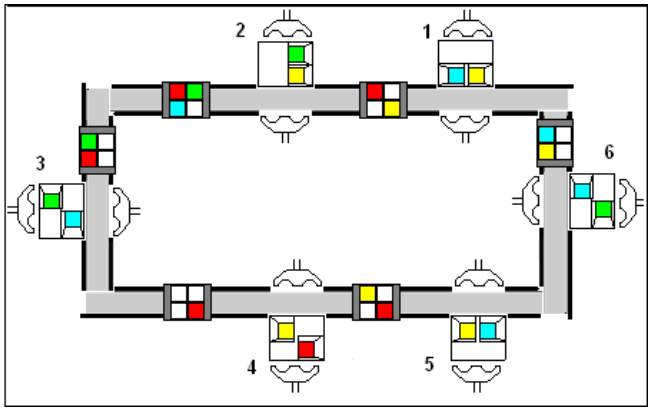


Figura 34: Proposta de manufatura de um sistema de montagem de peças.

Esta planta é concebida de forma que mecanicamente as estações possam ser retiradas ou inseridas, o que demonstrará a capacidade do sistemas SOA/MAS em adaptar-se a nova configurações.

Os paletes são identificados a cada vez que passarem por frente a uma estação de montagem, através de 3 sensores óticos em uma codificação binária (ex: acionado – desacionado – acionado = palete 5), assim, o *software* que modela o agente peça indicará aos demais agentes que seu modelo físico é o palete que possuir a codificação 5, e ao passar pela frente do agente de montagem, este irá identificar qual palete se encontra a sua frente.

Esta codificação poderia ser realizada por código de barras ou sensor magnético codificado (RFID), mas foi utilizado sensores óticos identificados em combinação binária, que apesar de limitado no número máximo de combinações distintas, tem o custo mais baixo e de fácil implementação. A Figura 35 mostra em detalhe os sensores e a codificação no palete.

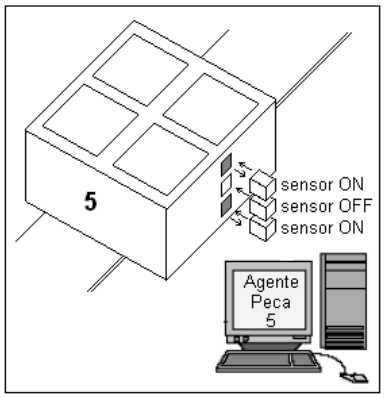


Figura 35: Palete com codificação binária por sensores.

O gerenciamento dos dispositivos atuadores (cilindros) e leitura dos sensores é realizado por um microcontrolador PIC16F877, que recebe comandos através de sua porta RS232 e realiza a sequência de acionamentos que promovam a montagem solicitada, bem como emite sinais de que manifestam o estado da montagem também por esta porta.

## 6.2 PROGRAMAÇÃO DO SISTEMA SOA/MAS EM JADE E FIPA

Partindo do sistema de manufatura proposto como demonstrador em 6.1, detecta-se que será necessária a implementação de agentes de recursos e agentes de produtos. Quanto aos agentes de transporte não serão implementados neste momento pois em análise prévia verifica-se que a esteira não tem funcionamento autônomo e não requer um gerenciamento de acionamento, ou seja, não haveria serviços que ela pudesse dispor.

Relacionando os agentes que irão compor a plataforma, temos:

a) Agente de recursos: agente de montagem, responsável pela montagem de cubos sobre o produto;
b) Agente de produto: agente peça, responsável pela busca e alocação dos serviços necessários para executar o fluxo de montagem requerido.

A Figura 36 apresenta os diagramas de classe dos agentes de montagem (agente de recurso) e dos agentes de peça (agente de produto).

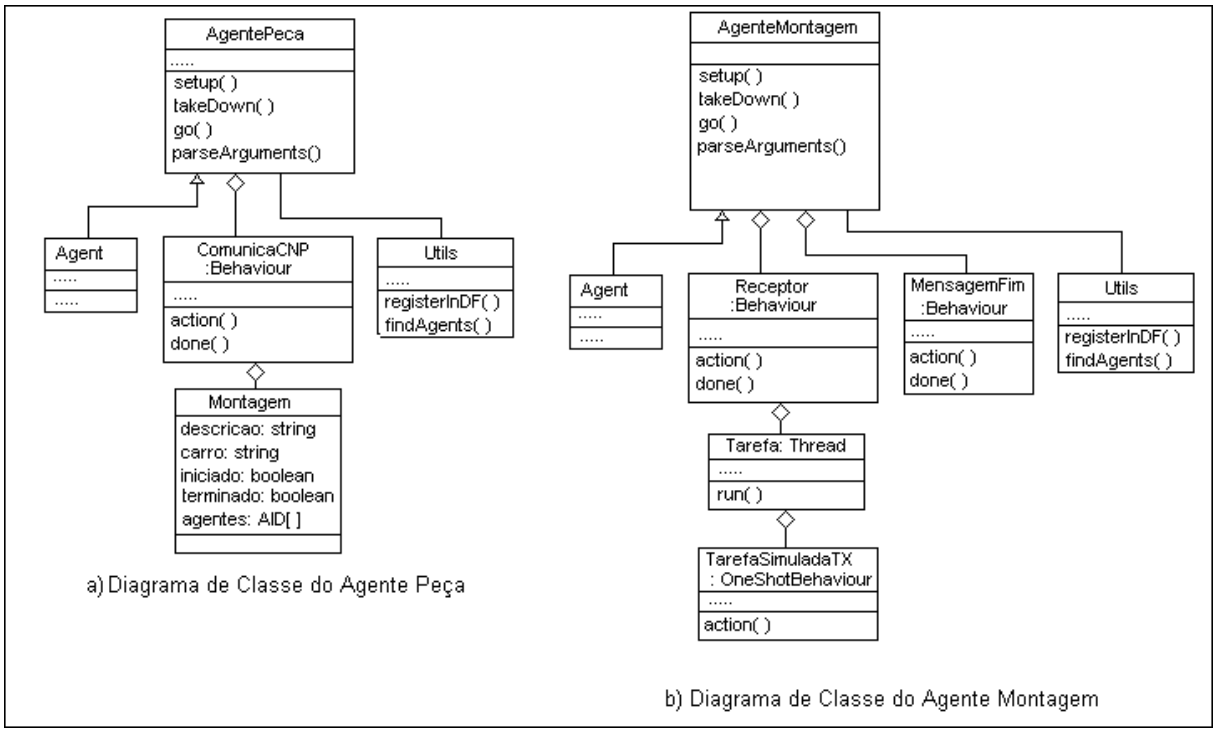


Figura 36: Diagrama de classe dos agentes de produto e agentes de recursos.

Ambos agentes são implementados a partir da classe *Agent*, disponível na plataforma JADE. Tanto o agente peça (PA) quanto o agente montagem (RA) utilizam uma classe chamada *Utils*, que fornece métodos para postagem dos serviços e pesquisa destes. A primeira ação do agente peça é buscar no DF a relação de agentes que realiza os serviços desejados e a partir deste ponto começam as negociações para alocar os recursos necessários. No agente de montagem destaca-se o comportamento TarefaSimuladaTX, que é o módulo de hardware e que fará o contato com os dispositivos de acionamento e sensoriamento propriamente dito.

A comunicação entre os agentes ocorrerá através da pilha TCP/IP/Ethernet em uma rede sem fio. É uma rede de fácil implementação, onde os IPs são fixos, de forma a serem descritos no momento de instanciar o agente. Por exemplo, ao criar uma instância do agente

um container necessita ser criado, para armazenamento deste e dos demais agentes que serão instanciados, e indicar onde se encontra o agente identificador (AMS), como no comando: *>java jade.Boot –container –host 192.168.1.214 Ag1:AgenteMontagem*. O comando *–host* está indicando que o AMS da plataforma encontra-se no endereço de IP 192.168.1.214.

Como forma de comunicação entre os agentes é utilizado o protocolo FIPA, através da utilização de dois protocolos alinhados, FIPA Contract Net e FIPA Request. Neste ponto a utilização do protocolo FIPA Contract Net para realizar toda a comunicação provocou um problema. A comunicação se dá com o agente iniciador solicitando um CFP (Call for Proposal) aos agentes participantes e aguarda mensagens de resposta, cujo ato performativo seja REFUSE ou PROPOSAL, quando então toma a decisão de qual agente será aceito para realizar a tarefa. O agente iniciador envia mensagens de REFUSE aos agentes rejeitados, liberando-os a negociar com outros agentes, e mensagens de ACCEPT_PROPOSAL para os agentes aceitos, e fica aguardando a mensagem de INFORM_DONE que indicará que o agente participante terminou a tarefa. Enquanto o agente participante esta realizando a tarefa, ele ignora eventuais mensagens recebidas de outros agentes. Este é um grande problema que ocorre, pois o tempo de tarefa pode ser elevado e outros agentes iniciadores ao contatarem este agente através de um CFP não saberiam seu estado (REFUSE ou PROPOSAL).

A solução implementada foi unir os dois protocolos. Desta forma, o agente peça busca o melhor agente para realizar sua tarefa através do protocolo *FIPA Contract Net*, ao encontrar encerra o protocolo e abre o protocolo *FIPA Request* somente com o agente vencedor. Com isto o agente vencedor está realizando a tarefa sobre o protocolo *FIPA* Request e pode responder a outros agentes peças que lhe solicitarem CFP via protocolo *FIPA Contract Net*, a fim de responder como ocupado (*REFUSAL*).

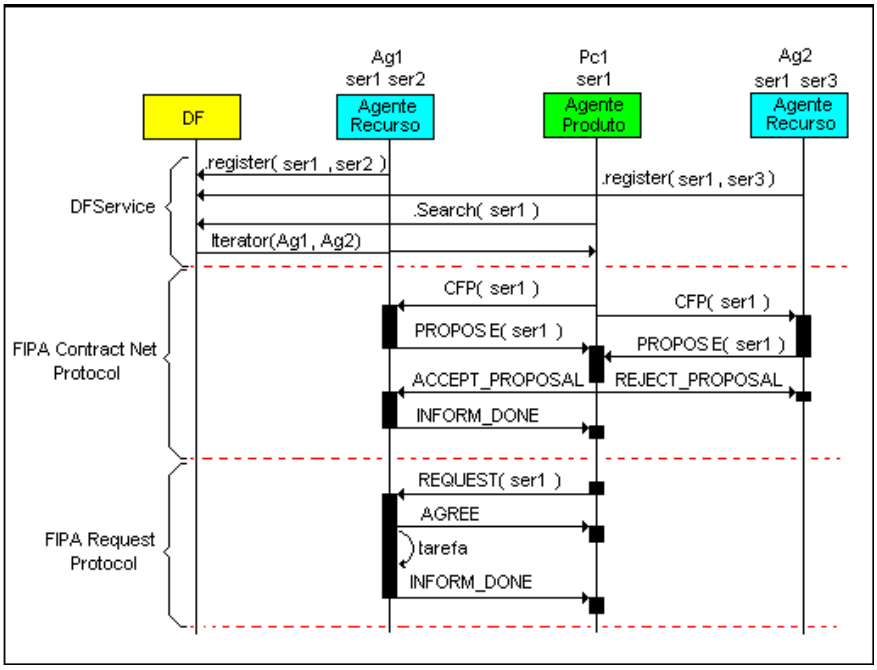


Figura 37: Diagrama de sequência UML com o protocolo de comunicação do sistema de montagem.

A Figura 37 apresenta o diagrama de sequência UML com a solução acima proposta. Nele observa-se que o processo começa com o agente Ag1 e Ag2 postando seus serviços no agente identificador de serviços (DF), respectivamente “ser1– ser2” e “ser1 – ser3”. Então o agente Pc1 solicita ao DF a lista de agentes capazes de realizar o serviço “ser1”, onde recebe na sequência a lista indicando os agentes Ag1 e Ag2. Começa então o protocolo *FIPA Contract Net*, onde o agente Pc1 envia um CFP (*Call for Proposal*) aos agentes Ag1 e Ag2, para que estes apresentem proposta para realizar o serviço “ser1”. Tanto o agente Ag1 quanto o agente Ag2 enviam suas propostas, que nesta implementação se resume a quantidades de tarefas concomitantes que o agente pode realizar, neste caso, somente a que foi solicitada. O agente Pc1 recebe as propostas pelo ato performativo PROPOSE e define qual agente é escolhido, através de critérios definidos. Nesta aplicação o critério é o maior número de serviços concomitantes e em seguida a proposta que chegou primeiro. Por este critério o agente escolhido foi o Ag1, fazendo com que o agente Pc1 envie um ACCEPT_PROPOSAL ao agente Ag1 e um REJECT_PROPOSAL, e fica aguardando um INFORM_DONE do

agente Ag1. Aqui se encerra o protocolo *FIPA Contract Net*, liberando o agente Ag2. Começa então o protocolo *FIPA Request* entre os agentes Pc1 e Ag1. O agente Pc1 envia um REQUEST para o agente Ag1 solicitando o serviço “ser1”, este informa ao agente Pc1 um AGREE indicando sua concordância. O agente Ag1 realiza a tarefa “ser1” e ao final envia um INFORM_DONE ao agente Pc1, indicando que a tarefa está terminada. Caso o agente Pc1 necessite de mais algum serviço, então processo começa novamente com a solicitação de busca no DF deste novo serviço e assim se sucede.

## 6.3 CONCEPÇÃO DE EXPERIMENTO DE VALIDAÇÃO

Uma das funcionalidades que se busca em sistemas de montagem é a modularidade e a facilidade de expansão. Esta última remete que o sistema possa ter um número crescente de módulos de montagem. A fim de que se estabeleça um universo definido para validação dos experimentos, optou-se por limitar o número de estação de montagem em 6 instâncias do agente de recursos, com habilidades definidas e o produto a ser montado com requisitos específicos, sendo até 6 instâncias do agente de produto. A Figura 38 apresenta a definição das estações de montagem e do produto a ser produzido como validação experimental.

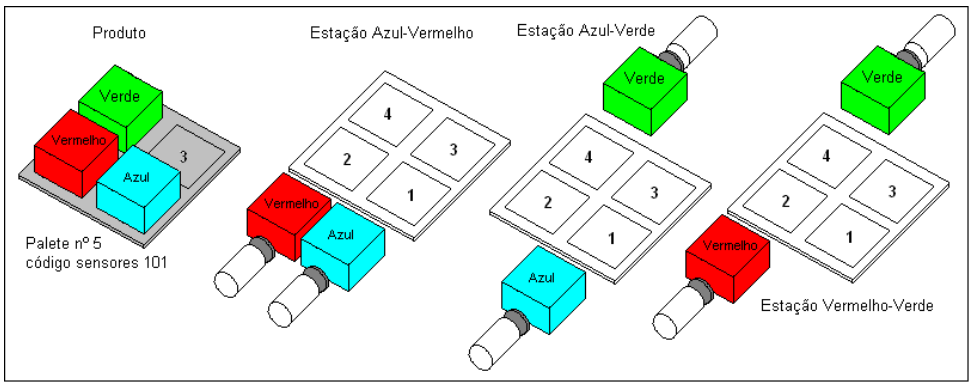


Figura 38: Definição do produto e estações a serem utilizados na validação experimental.

No projeto de experimento foram utilizados três tipos distintos de estações de montagem:

a) Estação A: Peças azuis na posição 1 e peças vermelhas na posição 2;

b) Estação B: Peças azuis na posição 1 e peças verdes na posição 4;

c) Estação C: Peças vermelhas na posição 1 e peças verdes na posição 4.

Já o produto que será validado é montado sobre o palete que se definiu como o de número 5 (código binário dos sensores 101), e requerido o produto como peça azul na posição 1, peça vermelha na posição 2 e peça verde na posição 4.

## 6.4 EMULAÇÃO EM SOFTWARE DO SISTEMA DEMOSTRADOR

Para simular as funcionalidades do sistema de transporte, das estações de montagem e dos paletes, em um sistema de montagem com arquitetura orientada a serviços utilizando sistemas multi-agentes, foi desenvolvido uma interface gráfica na linguagem Java, a qual contém classes e métodos que simulam os dispositivos físicos. Esta interface recebe o nome de agente monitor e é executado em um computador conectado à plataforma JADE, assim como cada computador que instancia os agentes de recurso, formando as estações de montagem. Este sistema virtual é apresentado na Figura 39, com o ambiente de simulação do agente monitor, que realizará a animação da simulação dos agentes de recursos, produto e da esteira de transporte.

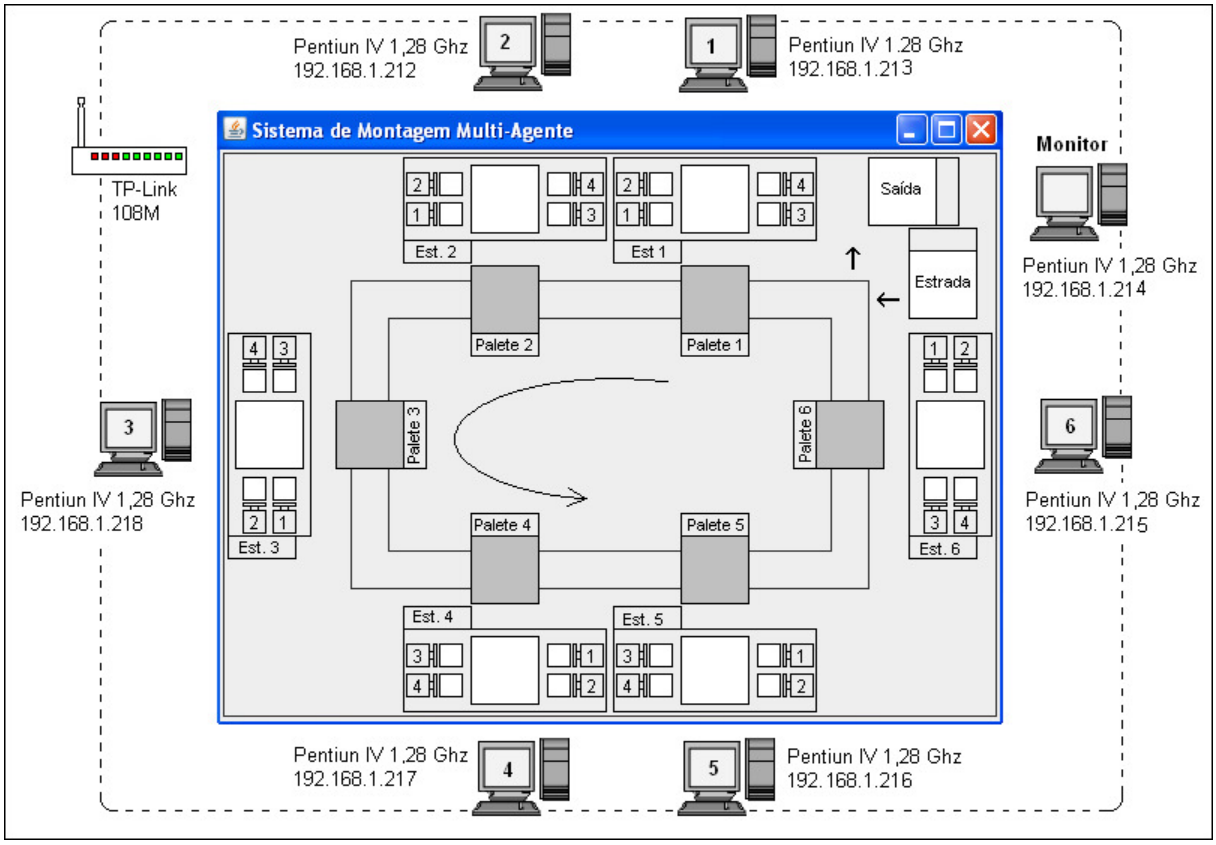


Figura 39: Sistema de montagem implementado em SOA/MAS.

A Figura 40 apresenta um diagrama de classe do agente monitor que implementa a interface de simulação gráfica e a comunicação dos demais agentes da plataforma, reconhecendo seus estados e simulando suas ações de forma gráfica.

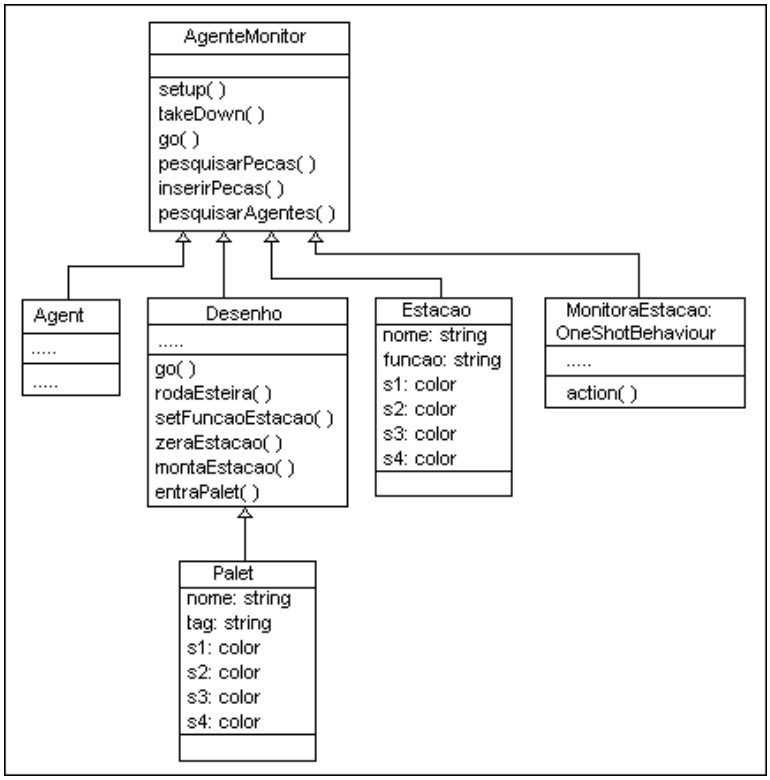


Figura 40: Diagrama de classes do agente monitor.

O agente monitor tem capacidade de simular até 6 estações de montagem e até 6 paletes em processo de montagem, delimitação esta em função do ambiente de simulação ficar muito carregado graficamente caso houvesse mais estações, o que prejudicaria a visualização. Pode-se acompanhar na Figura 41 a composição dos agentes do sistema com o agente monitor e os agentes DF e AMS.

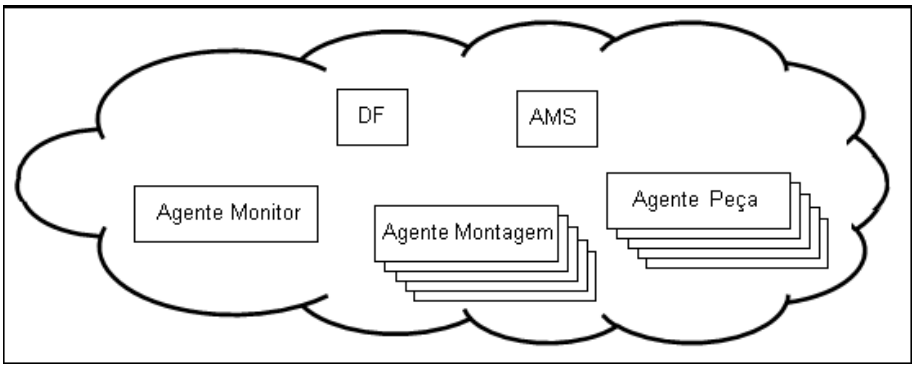


Figura 41: Composição dos agentes na implementação do sistema de manufatura.

Esse agente monitor pesquisa a cada segundo no *Agent Management System* (AMS) quais os agentes que estão no sistema e também no *Directory Facilitator* (DF) quais as habilidades dos agentes de montagem. Com isto o agente monitor consegue saber quais agentes estão no sistema, providenciando sua simulação. Já a execução da montagem acontece através da comunicação que ocorre entre os agentes de montagem e o agente monitor, onde o agente de montagem envia uma mensagem *REQUEST*, contendo o número do palete e as ações de montagem que devem ser realizadas. O agente monitor simula estas montagens e envia uma mensagem ao agente monitor que o solicitou com mensagem *CONFIRM*, indicando fim da montagem.

A plataforma JADE dispõe de um recurso de gerenciamento remoto de agentes (do inglês *Remote Management Agent* – RMA), que fornece um console gráfico para o controle e gerenciamento da plataforma (SILVA, 2003). Permite o controle dos estados relacionados com as diferentes etapas do ciclo de vida de cada agente em execução e serve como um controle principal para integração com outras ferramentas da plataforma JADE, tais como:

a) *Dummy Agent* (DA): é uma ferramenta gráfica de monitoramento e debugging para agentes em JADE. Mensagens em ACL podem ser criadas e enviadas, bem como pode-se obter uma listagem das mensagens ACL enviadas e recebidas com informações de seus atos performativos;
b) *Inspector Agent* (IA): É uma ferramenta que permite monitorar o ciclo de vida de um agente, suas trocas de mensagens e seus comportamentos que estão sendo executados. Permite o controle da execução do agente;
c) *DF Gui:* É uma ferramenta de interação com o DF do JADE. É possível criar uma rede de domínios e sub-domínios de páginas amarelas controladas, permitindo registrar, cancelar, modificar ou procurar agentes ou serviços.

d) *Sniffer Agent* (SA): SnifferAgent é uma ferramenta que fornece um diagrama de sequência UML entre determinados agentes, exibindo mensagens graficamente. Através desta ferramenta pode-se acompanhar as mensagens que estão sendo trocadas e a ordem com que acontece. É um recurso que permite análise dos protocolos que estão em comunicação dentro da plataforma.

A Figura 42 apresenta o diagrama de sequência UML resultante do processo de negociação entre os agentes, definido em um Sniffer Agent. Nesta figura pode-se perceber as consultas do agente peça (Pc1) ao *Directory Facilitator* (DF), as propostas e negociações com os agentes de montagem e também a comunicação do agente de montagem com o agente monitor (Monitor) para que este simule a operação do hardware.

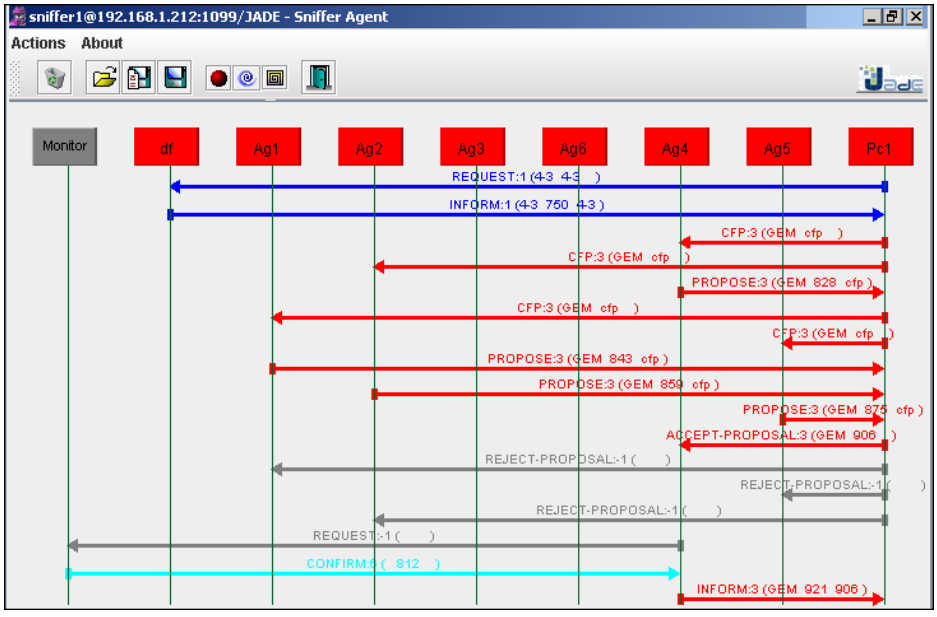


Figura 42: Diagrama de sequência UML da validação experimental, primeira etapa de montagem.

O diagrama em questão refere-se a uma solicitação de montagem de um palete com peças: 1-azul, 2-vermelha e 4-verde. Sendo que as estações possuem as habilidades: estação 1 (1-azul, 2-vermelha), estação 2 (1-azul, 4-verde), estação 3 (2-vermelha, 4-verde), estação 4 (1-azul, 2-vermelha), estação 5 (1-azul, 4-verde), estação 6 (2-vermelha, 4-verde). Nota-se

que inicialmente o agente peça (Pc1) envia uma solicitação de pesquisa de habilidades ao DF a fim de que identifique os agentes que executam a montagem de peça 1-azul, 2-vermelho e 4-verde. O agente peça (PC1) recebe do DF uma lista com a identificação dos agentes que podem executar a tarefa solicitada, seja integral ou parcialmente. Neste caso a resposta foi a seguinte lista: Ag1 que monta 1-azul e 2-vermelho, Ag2 que monta 1-azul e 4-verde, Ag4 que monta 1-azul e 2-vermelho e Ag5 que monta 1-azul e 4-verde. Nota-se que nenhum agente de montagem é capaz de executar integralmente o processo de montagem requisitado. De posse desta lista, o agente peça envia um convite à propostas (CFP) para os agentes que estão habilitados a executar sua tarefa, recebendo as propostas individualmente de cada agente de montagem. Nesta demonstração os agentes Ag2, Ag1 e Ag5 foram rejeitados, sendo o critério adotado para formação da escolha o fato do agente montar o maior quantidade de peças distintas na mesma operação. Caso exista mais de uma estação que atenda este critério, o agente peça selecionará aquela cuja mensagem de retorno seja recebida primeiro. Neste critério o agente Ag4 foi escolhido para executar a primeira parte da tarefa, montando as peças 1-azul e 2-vermelha, e a este agente foi mandado então uma mensagem de *ACCEPT_PROPOSAL* e aos demais agentes que participaram na negociação lhes é enviado uma mensagem de *REJECT_PROPOSAL*. O Ag4 agente de montagem faz uma solicitação ao agente monitor, enviando uma mensagem de *REQUEST*, a fim de solicitar a simulação da operação de montagem referida. Após executada a simulação da montagem o agente monitor AgMonitor responde ao agente de montagem Ag4 com uma mensagem CONFIRM, indicando que a montagem solicitada está encerrada.

Agora o agente peça possui montadas as peças 1-azul e 2-vermelho, necessitando agora a montagem da peça 4-verde. Então, como pode ser observado na Figura 43, o agente peça (PC1) envia uma solicitação de pesquisa de habilidades ao DF a fim de que este identifique os agentes que executam a montagem de peça 4-verde.

O agente peça (Pc1) recebe do DF uma lista com a identificação dos agentes que podem executar a tarefa solicitada. Neste caso a resposta foi a seguinte lista: Ag2 que monta 4-verde, Ag3 que monta 4-verde, Ag5 que monta 4-verde e Ag6 que monta 4-verde. Como só uma habilidade de montagem foi solicitada, então o critério que prevalece é o retorno de resposta mais rápida. E desta vez o agente Ag5 foi o contemplado, assim o agente peça Pc1 envia uma mensagem de *REJECT_PROPOSAL* aos agentes Ag2, Ag3 e Ag6, e envia uma mensagem de *ACCEPT_PROPOSAL* para o agente Ag5. Este, por sua vez, envia uma mensagem ao agente Monitor para solicitar a simulação da manutenção, pelo ato performativo *REQUEST*, aguardando o final da simulação para receber um sinal de CONFIRM, cabendo ao agente de montagem informar ao agente peça que a operação de montagem se encerrou, através do ato performativo *INFORM*.

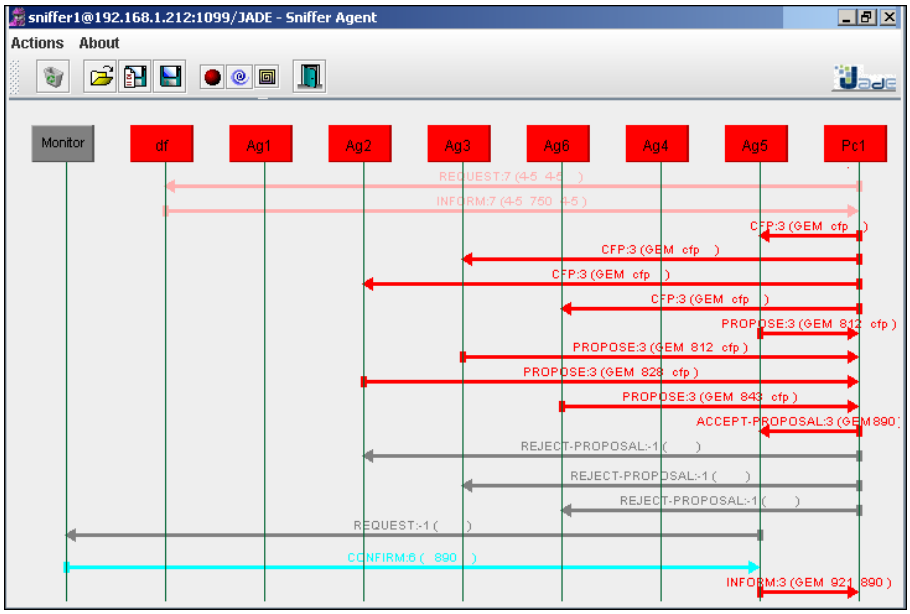


Figura 43: Diagrama de sequência UML da validação experimental, segunda etapa de montagem.

## 6.5 DESENVOLVIMENTO DO SISTEMA SOA/MAS PARA O DEMOSTRADOR

Seguindo as etapas propostas para implementação do sistema na arquitetura orientada a serviços e sistemas multi-agentes, tem-se:

1ª Etapa: identificação dos componentes que tem capacidade de funcionar autonomamente.

Na planta proposta do demonstrador em 6.1 identifica-se que cada estação de montagem tem capacidade de operar como um dispositivo autônomo, cuja funcionalidade é montar peças. Então, este é o serviço disponível aos demais agentes e que deverá ser registrado no DF. O sistema de transporte também tem capacidade de atuar autonomamente, assim considerado também como possível agente, cujo serviço é o transporte de peças de uma estação de montagem a outra. Tem-se então:

a) Agente de recursos: estações de montagem responsáveis pela montagem dos cubos nos paletes;
b) Agente de transporte: responsável pelo transporte dos produtos entre as estações de montagem.

2ª Etapa: verificar se os elementos de transporte devem ser mapeadas como agentes de transporte (TUA).

Aqui se observa que a tarefa de introduzir o produto da esteira transportadora para estação de montagem e sua volta para esteira ao final da manufatura é realizada pelos cilindros "*In*" e "*Out*" na própria estação de montagem, desonerando o sistema de transporte desta tarefa. E como a esteira alcança todas as estações de forma cíclica, então se definiu que esta aplicação não requer que o sistema de transporte seja modelado como agente, bastando garantir que a esteira esteja sempre ligada.

Então, neste modelo de manufatura não haverá agentes de transporte.

3ª Etapa: considerar cada produto ou conjunto de produtos como um agente de produto (PA).

Cada palete que entrar no sistema de montagem será considerado um agente, que solicitará os serviços aos demais agentes de montagem. Os agentes de montagem são componentes com *hardware* fixos ao longo da planta, e serão vinculados a uma rede (no momento da comunicação entre agentes). Já o palete, que também deverá se comunicar nesta rede, é um componente de *hardware* móvel ao longo da planta. Logo, os dispositivos que implementam estes agentes deveriam estar embarcados com o palete e deslocando-se ao longo da manufatura.

A solução implementada aqui é que o agente peça (palete) será modelado em *software* em um sistema computacional ligado ao sistema de manufatura, onde cada palete terá um número de identificação diferente.

Mas nesta implementação optou-se por simular o estado dos agentes de produto em software, assim, o módulo de *hardware* envia comando para um agente intitulado agente monitor, que simulará seu comportamento.

O primeiro comportamento a ser executado no agente de produto é um comportamento definido como *OneShotBehaviour*, que é executado uma única vez, e que nele é descrito um roteiro que visa buscar no agente identificador de serviços (DF) a lista dos agentes que têm capacidade de atender ao seu requisito de montagem. A partir de então ocorre a negociação entre os agentes para que os recursos sejam alocados.

4ª Etapa: promover a implementação dos dispositivos como agentes

A implementação em *software* dos agentes se dá utilizando a plataforma JADE, desenvolvida em JAVA, escolhida por dispor de classes para criar agentes em sistemas multi-agentes. Em JADE o primeiro agente a ser instanciado deve executar o comando *-gui*, que

instancia o agente de identificação AMS (*Agent Management System*), com função de identificar os agentes do sistema, e o agente de identificação de serviços DF (*Directory Facilitator*), responsável pela postagem e busca de serviços, como no exemplo de comando: *>java jade.Boot –gui Monitor:AgenteMonitor.*

Quanto aos agentes de recurso, este é implementado a partir de uma classe que a ferramenta JADE dispõe, sendo que os demais agentes são instâncias deste agente de recurso, que chamamos de agente de montagem. Um agente de montagem é implementado para dispor o serviço de montagem de peças.

Módulo de *hardware* dos agentes de montagem gerencia o acionamento dos dispositivos de montagem (cilindros) através do envio de comandos ao agente intitulado agente monitor, que recebe estes comandos e simula a execução da montagem dos cubos. Este dispositivo de *hardware* poderia ser um microcontrolador, um controlador lógico programável ou outro dispositivo similar, que receberia estes comandos e gerenciaria os acionamentos de cilindros e leitura dos sensores. Mas como a proposta é a simulação em *software* destes acionamentos, então isto é realizado por outro agente.

Nos agentes de montagem, a camada lógica também é descrita dentro dos comportamentos, e é responsável pelo envio de mensagens ao *hardware* e recebimento destas mensagens indicando o estado da montagem.

O módulo de comunicação é responsável pela negociação entre os agentes do sistema. Além da troca de informações entre o agente de produto e os agentes de recurso, também é necessário à postagem e busca dos serviços no *Directory Facilitator* – DF. Esta troca de mensagens é realizada pelo módulo de comunicação, utilizando o protocolo FIPA-ACL *Specification SC00037*. O protocolo FIPA se destaca por dispor de um conjunto de especificações, mensagens com rigorosa semântica.

A Figura 44 apresenta a formação do agente a partir do croqui do sistema de montagem.

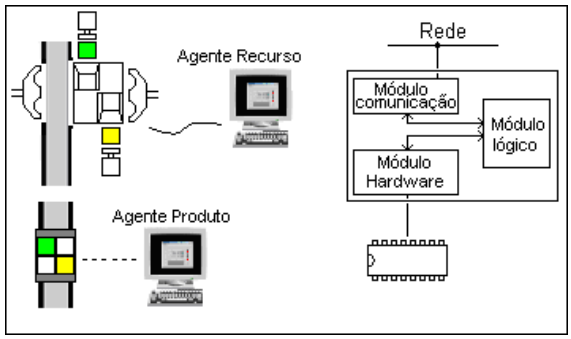


Figura 44: Dispositivos do sistema de manufatura considerado como agentes.

5ª Etapa: interligar os agentes em uma plataforma de comunicação entre agentes.

Para interligar os agentes utilizou-se o padrão FIPA, que dispõe de um conjunto de protocolos próprios para comunicação entre agentes, além de classes que facilitam a implementação. Seu código é escrito em JAVA e dentre os protocolos disponíveis optou-se por utilizar o *FIPA Request Interaction Protocol Specification (SC00026)* e o *FIPA Contract Net Interaction Protocol Specification (SC00026*).

Inicialmente os agentes de montagem postam seus serviços no DF e os agentes peças buscam o identificador ID dos agentes que realizam um serviço específico. Isto é realizado através de métodos disponíveis nas classes *DFAgentDescription* e *ServiceDescription*. A partir de então começa a negociação entre os agentes peça e os agentes de montagem, e então são utilizados os protocolos *FIPA Contract Net* e *FIPA Request*.

## 6.6 DESENVOLVIMENTO DO SISTEMA DE MANUFATURA PARA O DEMONSTRADOR USANDO IEC61131

A proposta deste trabalho visa desenvolver um sistema automatizado de manufatura no conceito de arquitetura orientada a serviços, utilizando recursos de sistemas multi-agentes. Mas para que se possa ter um paralelo entre um sistema tradicional com gerenciamento centrado no CLP e este sistema sob o conceito SOA e MAS, foi proposto implementar uma solução para o sistema de manufatura proposto em 6.1, utilizando o conceito de gerenciamento central em um CLP, a fim de comparar-se ambas implementações.

A simulação do sistema de montagem com o gerenciamento centrado em um CLP foi implementada através do *software* IsaGraf3.3, desenvolvido pela CJ International, que fornece uma interface gráfica que vincula os elementos a *flags* acessíveis pela linguagem de programação, que atende à norma IEC61131. Das cinco linguagens dispostas nesta norma optou-se pelo sequenciamento gráfico de funções (*Sequencial Function Chart*- SFC).

Pelos requisitos de produto concluiu-se que o sistema com estações de montagem alocadas ao redor de uma esteira circular, com acionamento contínuo, seria o mais viável para implementação da manufatura com gerenciamento centrado no CLP. A Figura 45 apresenta o ambiente de simulação proposto para um gerenciamento centrado em um controlador lógico programável.

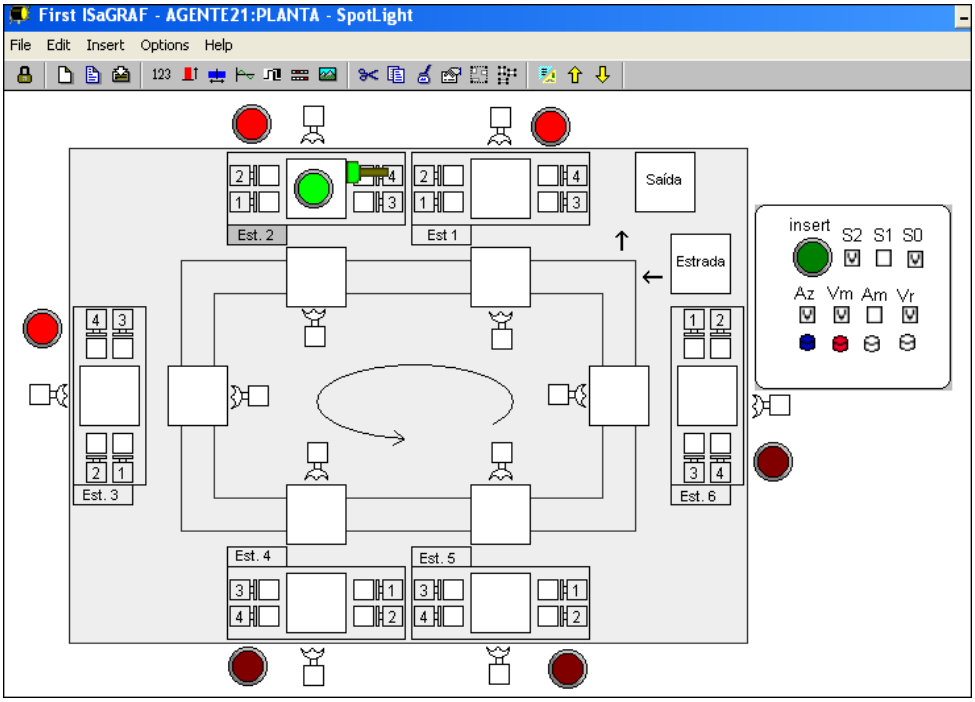


Figura 45: Planta didática virtual do sistema de montagem centrado no CLP.

Esta implementação é composta de uma interface gráfica que dispõe os componentes de *hardware*. Sobre cada componente há um objeto que simula o comportamento do componente, através da inserção de imagens vinculadas a uma variável. A solicitação de produção é realizada pela definição do tipo de montagem e pelo comando no botão de inserção. O tempo de movimentação da esteira foi definido em 1 segundo por passo, os movimentos dos cilindros que recolhem o palete e o devolvem à esteira foi ajustado para 2 segundos, enquanto os cilindros que fazem a montagem das peças estão ajustados para um tempo de avanço de 1 segundo.

A fim de garantir-se que possíveis desvantagens da versão CLP não fossem devido a uma programação inadequada, solicitou-se o apoio de um especialista em programação de CLP, que ministra aulas deste assunto em escola técnica, para desenvolvimento daquele programa.

A partir dos requisitos do produto inicial e croqui do sistema de montagem, este técnico definiu uma estratégia que consistiu em vincular os requisitos de montagem do palete

a *flags* específicas. Cada vez que o palete passa à frente da estação de montagem, esta verifica quais flags ainda não foram montadas e compara com sua capacidade de montagem. Caso haja alguma tarefa ainda não realizada e que possa ser realizado pela estação de montagem, então o palete é recolhido e a operação realizada, sendo então o palete devolvido à esteira, ajustando a *flag* para estado montada.

O programa poderia ser editado em linguagem de contatos (*ladder*), bastando uma transformação do código SFC para o *ladder*. Mas como o CLP que foi programado permitia a programação direta em SFC, esta foi a linguagem de programação escolhida. Cada estação possui um programa similar, o qual é apresentado na Figura 46 (esta figura também apresenta o croqui da estação de montagem). O croqui e programa em questão referem-se à estação 4, sendo a primeira transição ativada pela condição de chegada do palete através do sensor "est4" , e do código de palete número 5 (baseado nos valores dos sensores "S2", "S1", "S0"). Na sequência é verificado se a *flag* que ainda não foi montada coincide com sua capacidade de montagem, pela comparação entre as flags "az" e "Faz" e as *flags* "vm" e "Fvm". As flags "az" e "vm" dizem respeito à capacidade da estação montar peças azuis e vermelhas respectivamente. Já as flags "Faz" e "Fvm" dizem respeito à necessidade de montagem no palete.

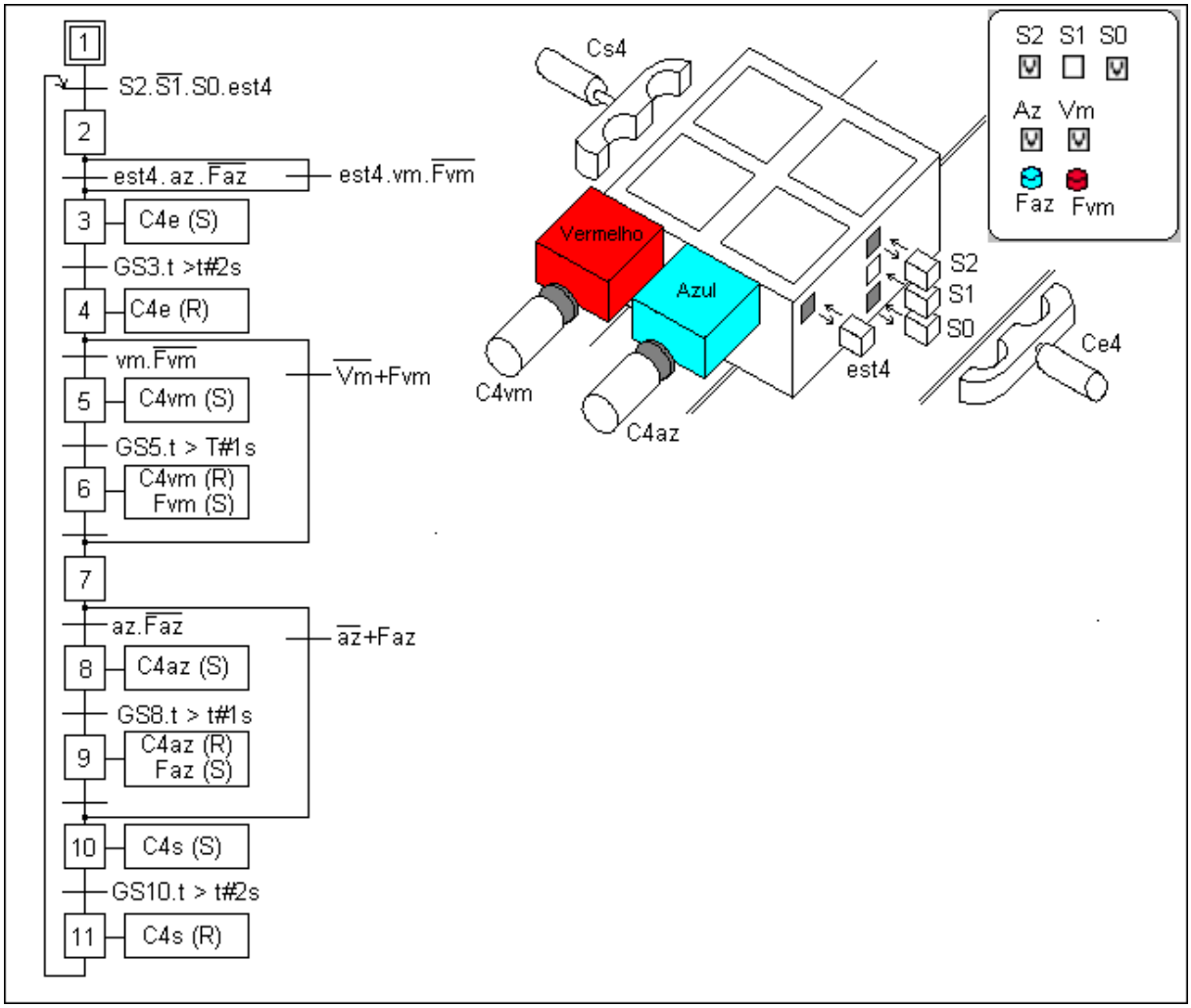


Figura 46: Diagrama SFC do programa do CLP.

# 7. RESULTADOS E ANÁLISES

A proposta conceitual visa conceber um sistema automatizado de manufatura sob uma arquitetura orientada a serviços, valendo-se de sistemas multi-agentes. O foco do trabalho é a forma com que o sistema é gerenciado, os comandos que ocorrem, a forma com que os agentes se comunicam e o resultado de suas negociações.

## 7.1 MÉTRICAS DE COMPARAÇÃO

As métricas dimensionam ou qualificam características de requisitos de um sistema, permitindo análise comparativa. Quando dimensionam diz-se que as métricas são quantitativas. Quando qualificam diz-se que as métricas são qualitativas. As métricas podem ser medidas diretamente (custo, esforço, linhas de código, velocidade de execução, memória, número de erros, etc.) ou indiretamente (funcionalidade, qualidade, complexidade, eficiência, confiabilidade, manutenibilidade, modularidade, etc.).

A proposta dessa dissertação aponta para aspectos mais qualitativos, que remetem ao desempenho do sistema (desempenho tende a ser qualitativo), o que implica que as métricas sejam de cunho qualitativo. Algumas métricas foram listadas e definidas como não pertinentes para a análise proposta, como sugere: custo (mais focado ao *hardware* do que ao *software* de um sistema produtivo), número de erros (diz mais respeito a sistemas que já estão em produção), confiabilidade (muito subjetivo para esta proposta), entre outros.

As métricas definidas como adequadas e pertinentes para a análise são apresentadas na tabela 4, junto com sua justificativa técnica, forma de obtenção e critério de avaliação.

Tabela 4: Relação de métricas utilizadas na validação experimental.

| **Métricas** | **Justificativa técnica** | **Forma de obtenção** | **Critério para avaliação** |
|---|---|---|---|
| Variação do Tempo de produção com aumento das estações. | O aumento de estações de produção não deve acarretar aumento no tempo de produção. | Montar o sistema com 1 estação e solicitar 1 produto, cronometrando a montagem. Repetir isto para 2, 3, 4, 5 e 6 estações. | O acréscimo no tempo de produção em função do aumento das estações deve ser o menor possível. |
| Manter a produção com a retirada de estações. | Em caso de falha em uma estação, o sistema de montagem deve continuar o processo, cabendo a outra estação absorver a tarefa faltante. | Montar o sistema com estações em que 2 delas tem a mesma habilidade. Requerer esta habilidade no produto e, durante o processo, desligar uma das estações. | Sendo o sistema modular, saindo uma das estações,outra deve assumir a tarefa, admitindo ter a mesma funcionalidade, sem requerer alteração na programação. |
| Manter a produção com a inserção de estações. | Em caso de formação de gargalo, o sistema deve permitir que mais estações sejam inseridas em auxílio ao sistema de montagem. | Montar o sistema com estações com habilidades distintas. Requerer produto e, durante o processo, inserir mais uma estação com a mesma habilidade. | O sistema deve permitir expansão. A inserção de uma estação não deve requerer alterações na programação, nem demandar uma parada do sistema. |
| Quantidade de memória ocupada pelo programa somando todo o sistema. | A quantidade de memória ocupada define a necessidade de capacidade computacional do sistema para implantação. | Montar o sistema com 6 estações e solicitar a montagem de 1 produto. Monitorar o tamanho da memória ocupada. | Quanto menor a ocupação de memória melhor será seu custo em relação aos equipamentos. |
| Competências necessárias para operar e implementar as alterações no sistema de montagem. | Maior complexidade na implementação de alterações remete a necessidade de conhecimentos específicos do operador/programador do sistema. | Propor mudanças no sistema de montagem e identificar as capacidades necessárias ao operador para implementar a modificação. | Quanto menor o conhecimento requerido, mais operacional o programa torna-se quando aplicado em produção industrial. |
| Linhas de código de programação. | O número de linhas de código de um programa denota o esforço de programação. | Contagem das linhas de código ou elementos lógicos inseridos. | O número de linhas de código deve ser o menor possível. |

### 7.1.1 Métrica “tempo de produção com aumento das estações”

Como plano de teste para obtenção dos dados nos sistemas foram inseridas estações de montagem com habilidade de montar peças azuis na posição 1 e vermelhas na posição 2. Inicialmente foi inserida 1 estação e solicitado a montagem de peça, sendo azul na posição 1 e vermelho na posição 2, sendo cronometrado o tempo de montagem. A Figura 47 apresenta o sistema de montagem utilizado para analisar esta métrica.

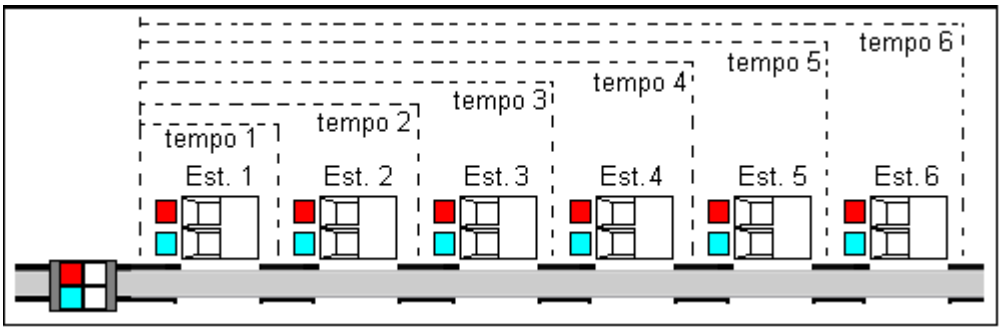


Figura 47: Sistema de montagem com 6 estações para análise do tempo de montagem com o acréscimo de estações.

O mesmo processo foi repetido para o sistema com 2 a 6 estações de montagem, todas com as mesmas habilidades.

Como método de análise foi realizado uma análise de experimento fatorial completo (MONTGOMERY, 2001), onde a variável controlada (VC) é o número de estações, a variável de resposta (VR) é o tempo de manufatura, sendo realizada a 6 níveis.

A determinação do número de repetições que deve ocorrer, amostra, é definida executando um experimento arbitrário como referência. Optou-se então por realizar o experimento com gerenciamento em SOA e 1 estação de montagem, sendo realizado 5 cronometragens, número este arbitrário. A tabela 5 apresenta o resultado deste experimento inicial.

Tabela 5: Tempo de produção com 1 estação e 5 amostras no gerenciamento em SOA.

| **Tempo de montagem (em s)** | Tempo médio (em s) |
|---|---|
| 26,94 | 26,72 |
| 26,68 | |
| 26,80 | |
| 26,62 | |
| 26,50 | |

O número de amostras estimado pode ser definido pela equação 1:

$$n = \frac{t^2 . S^2}{E^2} \tag{1}$$

Onde:

n = número de amostra estimado, arredondado para o número inteiro superior;

t = fator de Student, definido pelo grau de liberdade e índice de confiabilidade;

$S^2$ = variância estimada do desvio padrão;

E = Variabilidade máxima aceitável, definida como maior valor de variabilidade nas amostras.

Sendo o índice de confiabilidade de 95% e o grau de liberdade de 4 ($n_{estimada} - 1$), o valor de t = 2,13. Assim a definição da amostra é apresendado na equação 2.

$$n = \frac{(2,13)^2 . 0,002634}{(0,19)^2} = 2,16 \tag{2}$$

Assumindo o valor inteiro superior, o número mínimo de amostras necessário será de 3. Como prática para todos os experimentos foi definido que o número de amostragem será de 5 cronometragens.

A tabela 6 apresenta os resultados para o sistema com gerenciamento centrado no CLP e a tabela 7 apresenta os resultados para o sistema com gerenciamento utilizando o conceito de SOA/MAS.

Tabela 6: Tempo de produção com aumento das estações no gerenciamento com CLP.

| **Número de estações** | **Tempo de montagem (em s)** | | | | | **Tempo Médio (em s)** |
|---|---|---|---|---|---|---|
| 1 | 13,812 | 13,765 | 13,812 | 13,813 | 13,766 | 13,794 |
| 2 | 13,812 | 13,812 | 13,812 | 13,781 | 13,812 | 13,806 |
| 3 | 13,813 | 13,813 | 13,812 | 13,812 | 13,812 | 13,812 |
| 4 | 13,812 | 13,812 | 13,812 | 13,812 | 13,813 | 13,812 |
| 5 | 13,813 | 13,812 | 13,812 | 13,813 | 13,812 | 13,812 |
| 6 | 13,812 | 13,813 | 13,812 | 13,812 | 13,812 | 13,812 |

Obs: computador Intel Aton 1,6GHz, 1GB RAM, SO WinXP SP3.

Tabela 7: Tempo de produção com aumento das estações no gerenciamento em SOA.

| **Número de estações** | **Tempo de montagem (em s)** | | | | | **Tempo Médio (em s)** |
|---|---|---|---|---|---|---|
| 1 | 26,99 | 26,68 | 26,80 | 26,62 | 26,50 | 26,72 |
| 2 | 28,69 | 26,94 | 27,71 | 26,87 | 26,43 | 27,33 |
| 3 | 28,06 | 29,25 | 29,50 | 27,31 | 28,19 | 28,26 |
| 4 | 29,37 | 30,75 | 27,37 | 28,06 | 28,55 | 28,82 |
| 5 | 31,31 | 29,56 | 29,07 | 27,87 | 31,55 | 29,87 |
| 6 | 35,12 | 30,87 | 30,80 | 30,55 | 28,18 | 31,10 |

Obs: computadores Intel Aton 1,6GHz, 1GB RAM, SO WinXP SP3.

A análise realizada com esta métrica é se a inserção de estações é significante em termos de tempo de produção.

Em (MONTGOMERY, 2001) descreve análise de variância através de uma tabela ANOVA, onde se pode determinar a significância de uma variável controlada (número de estações) a partir de valores de uma variável de resposta (tempo de produção). As Tabelas 8 e 9 representam a tabela ANOVA da análise do projeto fatorial completo do sistema gerenciado por CLP e gerenciado em SOA, respectivamente.

Tabela 8: Tabela ANOVA do experimento do tempo de produção com aumento das estações, no gerenciamento no CLP.

| Fator | SQ | GDL | MQ | Fo | | $Ft_{(0,95;n1;n2)}$ |
|---|---|---|---|---|---|---|
| A | 0,0014 | 5 | 0,000286 | 2,02 | < | 2,62 |
| Erro | 0,0034 | 24 | 0,000142 | | | |
| Total | 0,0048 | 29 | | | | |

Tabela 9: Tabela ANOVA do experimento do tempo de produção com aumento das estações, no gerenciamento em SOA.

| Fator | SQ | GDL | MQ | Fo | | $Ft_{(0,95;n1;n2)}$ |
|---|---|---|---|---|---|---|
| A | 65,16 | 5 | 13,03 | 6,50 | > | 2,62 |
| Erro | 48,11 | 24 | 2,00 | | | |
| Total | 113,28 | 29 | | | | |

Na Tabela 8, para o experimento com o sistema gerenciado no CLP, o parâmetro Fo sendo menor do que o parâmetro Ft, para uma confiabilidade de 95%, indica que o fator A (número de estações) não é significativo. Já na tabela 9, com sistema gerenciado em SOA, o valor do parâmetro Fo é maior do que o valor de Ft, para mesma confiabilidade, e isto define que o fator A neste caso é significativo.

Para esta métrica percebe-se que num sistema com processamento centrado no CLP o acréscimo de estações não é significativo ao tempo de produção. Mas no sistema com arquitetura orientada a serviços o acréscimo de estações é significativo para o tempo de produção. Isto se explica pelo fato de que no CLP a lógica já está descrita, enquanto que em SOA ocorrerá uma negociação entre estações, e quanto maior for o número de estações mais negociações ocorrerão, logo, maior o tempo dispensado.

### 7.1.2 Métrica “manter a produção com a retirada de estações”

Como plano de teste para obtenção dos dados, o sistema de montagem foi configurado para 3 estações de montagem, ou seja, 3 instâncias do agente de recursos. Estação 1 com habilidade de montar peças azuis na posição 1 e vermelha na posição 2, estação 2 com

habilidade de montar peças azuis na posição 1 e verde na posição 4 e estação 3 com habilidade de montar peças vermelhas na posição 2 e verde na posição 4. Então foi solicitado montagem de um produto com peça azul na posição 1, peça vermelha na posição 2 e peça verde na posição 4. A Figura 48 apresenta o sistema de montagem utilizado para analisar esta métrica.

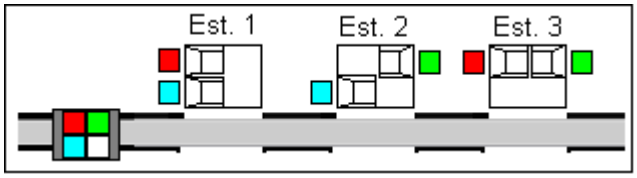


Figura 48: Sistema de montagem com 3 estações para análise do comportamento com a retirada de estações.

Esta mesma montagem fora solicitada ao sistema por 3 vezes, onde em cada solicitação uma estação de montagem fora retirada do sistema. A tabela 10 apresenta os resultados dos efeitos observados.

Tabela 10: Efeito no sistema com a retirada de estações.

| Estação retirada | SOA | CLP |
|---|---|---|
| 1 | O sistema passou a ser montado pelas estações 2 e 3. Não houve necessidade de reprogramação. | O sistema passou a ser montado pelas estações 2 e 3. Não houve necessidade de reprogramação. |
| 2 | O sistema passou a ser montado pelas estações 1 e 3. Não houve necessidade de reprogramação. | O sistema passou a ser montado pelas estações 1 e 3. Não houve necessidade de reprogramação. |
| 3 | O sistema passou a ser montado pelas estações 1 e 2. Não houve necessidade de reprogramação. | O sistema passou a ser montado pelas estações 1 e 2. Não houve necessidade de reprogramação. |

No CLP a retirada da estação se deu pela inibição dos sensores que detectam a presença do palete em frente à estação de montagem. Em SOA a retirada da estação se deu pelo fechamento da instância do agente de montagem.

Conclui-se então que a retirada de estações em ambos sistemas não prejudica a continuidade de produção, desde que outras estações tenham habilidades idênticas às da estação que saiu.

### 7.1.3 Métrica "manter a produção com a inserção de estações"

Como plano de teste para analisar esta métrica foi utilizado os mesmos sistemas de montagem propostos na Figura 48. Só que agora o mesmo requisito de produto foi solicitado por uma sequência de vezes, de forma a formar um gargalo de produção, até que houvesse mais solicitantes dos que estações para montagem. A inserção de mais uma estação similar à estação 1 é implementada, com o nome de estação 4, e o comportamento do sistema é observado. Os resultados são apresentados na tabela 11.

Tabela 11: Efeito no sistema com a inserção de estações.

| Estação inserida | SOA | CLP |
|---|---|---|
| 4 | As montagens que necessitam da estação 1 passaram também a ser encaminhadas para a estação 4, diminuindo o gargalo. Não houve necessidade de reprogramação. | As montagens que necessitam da estação 1 passaram a serem encaminhadas para estação 4 quando esta estava ocupada, diminuindo o gargalo. Houve a necessidade de parada do sistema e reprogramação do CLP, inserindo um trecho de programa com lógica semelhante ao da estação 1. |

Aqui surge uma das funcionalidades prometidas pela arquitetura orientada a serviços, utilizando sistemas multi-agentes, e que se comprova: a adaptabilidade A capacidade do sistema se expandir sem que haja necessidade de parar o sistema, tão pouco reprogramá-lo.

No sistema centrado no CLP a inserção de mais uma estação, apesar de ter sua lógica muito parecida com a lógica das outras estações, requeriu que o sistema fosse desligado para que a reprogramação do CLP ocorresse.

### 7.1.4 Métrica "quantidade de memória ocupada pelo programa"

Como plano de teste para obtenção dos dados dos sistemas foi utilizada a mesma configuração da Figura 47. A estação inserida tem as mesmas habilidades (montar peça azul na posição 1 e peça vermelha na posição 2). O produto solicitado foi uma montagem com peça azul na posição 1 e peça vermelha na posição 2.

Entre a solicitação da montagem e sua realização total foi monitorada a quantidade de memória ocupada, através do gerenciador de tarefas do sistema operacional. Fora considerado o pico de memória ocupada e no caso de sistema com mais de um computador foi considerado a soma das quantidades de memória ocupadas. Ao todo seis ensaios foram realizados, a cada ensaio uma estação a mais foi acrescentada, começando com uma estação de montagem e terminando com seis. A tabela 12 apresenta os resultados desta análise.

Tabela 12: Quantidade de memória ocupada para um processo de montagem em função do acréscimo de estações.

| | **CLP** | | **SOA** | |
|---|---|---|---|---|
| **Estação inserida** | **memória (em bytes)** | **normalizada** | **memória (em bytes)** | **normalizada** |
| 1 | 2.331.622 | 1,0000 | 87.004 | 1,0000 |
| 2 | 2.349.510 | 1,0077 | 90.020 | 1,0346 |
| 3 | 2.365.781 | 1,0146 | 100.444 | 1,1544 |
| 4 | 2.381.263 | 1,0213 | 119.040 | 1,3682 |
| 5 | 2.396.824 | 1,0280 | 127.712 | 1,4678 |
| 6 | 2.412.464 | 1,0346 | 158.200 | 1,8183 |

Similar à métrica de tempo de montagem, essa métrica também sofre influência da ferramenta de desenvolvimento e simulação. Para que esse fator não se sobreponha à métrica

que se quer analisar, que é a quantidade de memória que o sistema ocupa com o acréscimo de estações, então se optou por normalizar esta quantidade sobre um valor de memória requerido para montagem com uma estação. Essa normalização fornece um dado mais adequado para análise, e é determinada pela equação 3.

$$\text{Memória normalizada com n estação} = \frac{\text{memória com n estações}}{\text{memória com 1 estação}} \tag{3}$$

Percebe-se nesta análise que no sistema em que o gerenciamento é centrado em um CLP há pouco incremento de memória em função do aumento no número de estações. Numa variação de estações de 1 a 6 o acréscimo de memória ocupada não chega a 4%.

Já no sistema em que o gerenciamento segue o princípio de SOA, o acréscimo de estações implica num acréscimo significativo na ocupação de memória. Na variação de 1 a 6 estações o acréscimo de memória chega próximo a 82%.

### 7.1.5 Métrica "competências necessárias ao operador do sistema de montagem"

Esta talvez seja a métrica mais subjetiva e ao mesmo tempo uma das mais importantes, pois reflete diretamente na pessoa que irá operar o sistema. Um sistema de manufatura pode requerer alto grau de conhecimentos das pessoas que o desenvolve, pois seu custo só impacta durante o tempo de desenvolvimento, sendo diluído no decorrer da produção, quando este não mais participa. Mas não deve requerer competências muito complexas para sua operação, pois o operador do sistema irá trabalhar continuamente com este e seu custo será diretamente agregado a produção. Competências mais complexas correspondem a um custo mais elevado.

Como plano de teste foram listadas as ações que pertinentes ao operador do sistema, que já ocorrera na análise das métricas anteriores, e julgado as competências técnicas que este

operador requer à atividade. A tabela 13 apresenta as atividades e as competências requeridas em ambos sistemas.

Tabela 13: Competências requeridas para operar o sistema de montagem.

| Atividades | SOA | CLP |
|---|---|---|
| Ligar e preparar o sistema de montagem para o trabalho. | operar sistemas industriais; operar sistema operacional de computador; | operar sistemas industriais; |
| Solicitar montagens de produtos. | operar sistemas industriais; operar sistema operacional de computador; | operar sistemas industriais; |
| Retirar estações de montagem no sistema. | operar sistemas industriais; operar sistema operacional de computador; | operar sistemas industriais; |
| Inserir estações de montagem no sistema. | operar sistemas industriais; operar sistema operacional de computador; | operar sistemas industriais; operar sistema operacional de computador; programar controladores lógicos. |

Nesta análise nota-se que para ligar, preparar e solicitar produção, o sistema de gerenciamento centrado no CLP requer apenas que o operador tenha competências para operar sistemas industriais, haja vista que o CLP já possui interfaces de comunicação com o operador. Ao passo que para estas tarefas um sistema de montagem em SOA necessita que o operador tenha capacidade de utilizar o sistema operacional, pois ali estão os comandos para instanciar os agentes.

Mas a grande diferença é quando o sistema requer a inserção de mais estações, em que o sistema de montagem em SOA demanda o mesmo operador, e o sistema de montagem centrado no CLP demanda um programador de controlador lógico programável.

Assim, conclui-se que para esta métrica o sistema de montagem centrado no CLP requer, no conjunto das atividades, um operador mais qualificado do que um sistema de montagem em SOA.

### 7.1.6 Métrica “linhas de código de programação”

Esta métrica está diretamente relacionada ao esforço de programação. Quanto maior o número de linhas de código, maior é o esforço de programação requerido.

O sistema de automação da manufatura implantado no conceito SOA/MAS foi desenvolvido utilizando a linguagem de programação JAVA, sendo direta a determinação do número de linhas de código. Já o sistemas cujo gerenciamento é por CLP utilizou a linguagem SFC, que se baseia em componentes lógicos de forma gráfica. Neste caso, o dimensionamento se dará pelo número de componentes lógicos utilizados, valendo-se de suas linhas de comando.

Em ambas implementações foi utilizado uma interface gráfica para exibir o resultado da simulação, porém esta interface não afeta diretamente a implementação, apenas simula seu estado para que se possa visualizar o estado da manufatura, assim, os códigos destas interfaces não é considerado para análise desta métrica, assim tem-se o esforço para programar a manufatura. A tabela 14 apresenta os resultados desta análise.

Tabela 14: Linhas de código de programação.

| **Elementos no sistema** | **SOA linhas de código** | **CLP Elementos lógicos e comandos** |
|---|---|---|
| 1 estação e 1 palete | 620 | 58 |
| 2 estação e 1 palete | 620 | 94 |
| 3 estação e 1 palete | 620 | 130 |
| 4 estação e 1 palete | 620 | 166 |
| 5 estação e 1 palete | 620 | 202 |
| 6 estação e 1 palete | 620 | 238 |

Aqui se percebe um diferencial ente os dois conceitos. Independente do número de linhas de código ser absoluto, o que depende da linguagem em uso, nota-se que no conceito SOA uma vez emanado o esforço de programação para 1 estação, o esforço é o mesmo para

mais de 1 estação, pois se trata de instâncias deste mesmo código que serão executados. Já no conceito centrado no CLP a cada nova estação há um acréscimo de esforço de programação por parte do programador, e este aumento é crescente.

## 7.2 ANÁLISE DOS RESULTADOS.

Analisando o resultado de cada métrica que fora observada, pode-se atribuir vantagens de desvantagens a cada um dos conceitos, valendo-se da comparação. A Tabela 15 descreve melhor esta análise sobre todas as métricas analisadas.

Tabela 15: Análise das métricas observadas.

| **Métrica** | **Análise** | **Sistema de melhor desempenho** |
|---|---|---|
| Variação do Tempo de produção com aumento das estações | Em CLP o acréscimo de estações não é significativo para o tempo de produção, diferente do SOA, onde o acréscimo de estações afeta o tempo de produção. | CLP |
| Manter a produção com a retirada de estações | A retirada de estações em ambos sistemas não prejudica a continuidade de produção, desde que outras estações tenham habilidades idênticas às da estação que saiu. | Desempenho igual. |
| Manter a produção com a inserção de estações | No CLP para inserção de uma nova estação foi necessário desligar e reprogramar novamente. Já no SOA bastou instanciar a nova estação. | SOA |
| Quantidade de memória ocupada pelo programa | Com CLP o acréscimo de estações pouco interferiu na quantidade de memória ocupada (4% de 1 para 6 estações). Já em SOA há uma diferença bem mais significativa (82% de 1 para 6 estações). | CLP |
| Competências necessárias ao operador do sistema de montagem | Em SOA, a operação do sistema requer conhecimentos básicos de sistema operacional, mesmo para inserção e retirada de estações. No CLP é necessário um programador para realizar as alterações no sistema. | SOA |
| Linhas de código de programação. | Em SOA, por se tratar de instâncias, o esforço de programação é único, independente do número de estação. Já em CLP a cada inserção de estação torna-se necessário acrescentar códigos, onerando mais esforço de programação. | SOA |

Não se trata aqui de definir qual sistema é melhor, mas indicar qual sistema responde melhor para uma determinada métrica. A aplicação é que definirá se uns requisitos de métrica serão mais importantes do que outros.

# CONCLUSÕES E TRABALHOS FUTUROS

No cenário da manufatura industrial, a necessidade de produtos altamente personalizados remete à busca de novos paradigmas para gerenciamento dos sistemas de automação da produção. Mas isto não significa o desuso dos paradigmas anteriores, visto que estes são ainda encontrados em implementações, com suas características e funcionalidades.

Neste contexto a arquitetura orientada a serviços utilizando sistemas multi-agentes aplicados em manufatura industrial se propõe a ser mais uma alternativa de gerenciamento de dispositivos de automação da produção, oferecendo funcionalidades mais adequadas à diversificação de produção, sem o propósito de substituir outras tecnológicas.

A utilização da plataforma JADE na implementação em JAVA trouxe ao sistema multi-agente a estrutura de comunicação entre os dispositivos, e uma linguagem estruturada, universalmente conhecida. Porém, o meio físico de comunicação e, principalmente, a tecnologia do sistema que irá suportar a programação do agente, é um empecilho na realidade de chão de fábrica. A necessidade de capacidade de processamento do dispositivo para que suporte a execução de um agente é alta, em nível de um sistema operacional, o que fica difícil de embarcar tal tecnologia em dispositivos de automação. Outro fator é a maturidade tecnológica dos sistemas operacionais vigentes frente aos requisitos de chão de fábrica. Não há sistemas operacionais robustos capazes de garantir a continuidade do processamento de sinais nas condições que os dispositivos de automação são submetidos na linha de produção. Poucos são os hardware que suportam sistemas operacionais com características peculiares ao chão de fábrica, como os PCs industriais. O que não ocorre com os controladores lógicos programáveis, que já possuem características melhores desenvolvidas para suportar o ambiente de fábrica.

Apesar de pouco observada na manufatura industrial, os estudos de caso mostraram aplicações industriais da arquitetura orientada a serviços e dos sistemas multi-agentes. Mesmo

havendo diferenças conceituais, este trabalho provou que ambos podem ser complementares, e ser implementados em conjunto para formar um sistema de manufatura. O SOA contribuiu com o conceito dos componentes que dispõe e requisitam serviços, e o MAS contribuiu com a forma ativa com que os agentes buscam comunicar-se, visando a solução do problema de manufatura.

Mesmo sendo um sistema promissor, durante a validação experimental observou-se que em algumas métricas o sistema em SOA/MAS deixou a desejar, como no significativo aumento do tempo de produção com um aumento do número de estações ou no acréscimo de memória necessário a cada acréscimo de agente. Mas o mais significante é a métrica que trata da inserção e retirada de estações, onde o sistema gerenciado por controlador lógico programável requer um operador com domínio de programação em CLP, afim de que possa editar e compilar um novo programa. Ao passo que em sistemas gerenciados em SOA o operador necessita conhecer os comandos para instanciar os agentes.

A validação experimental focou sistemas com no máximo 6 estações, mas pode-se concluir, por análise, que para um sistema em SOA com mais de 6 estações estaria limitado somente pela capacidade de comunicação do meio físico da rede. Já no CLP a limitação estaria na capacidade de memória do CLP, e isto é um requisito que define o custo deste.

A proposta de abordagem de um sistema de manufatura com gerenciamento em SOA/MAS mostrou-se eficaz, e mostra o caminho para que este processo seja realizado em outros dispositivos ligados à manufatura industrial.

Durante a validação experimental foram utilizadas plantas virtuais para simulação da manufatura. A proposta para trabalho futuro é implementar o sistema de montagem e transporte real, a fim de que se torne uma planta didática, servindo para estudos, ensaios e demonstrações de arquiteturas orientadas a serviços utilizando sistemas multi-agentes, com

cunho didático-pedagógico, trazendo a conotação real do processo produtivo e suas peculiaridades que aqui neste trabalho foram descartadas.

Na visão acadêmica os conceitos de sistemas distribuídos, arquiteturas orientadas a serviços e sistemas multi-agentes são fácilmente assimilados em função de estudos e perquisas realizadas nesta área. Mas na visão industrial, do empresário na indústria, estes conceitos geram dúvidas quanto a possibilidade de mudança de paradigma de gerenciamento da sua manufatura. E neste foco que a comparação do gerenciamento com arquitetura orientada a serviços utilizando sistemas multi-agentes com o sistema de gerenciamento mais comum, centrado no CLP, tráz um bom argumento para o convencimento do mercado industrial, que normalmente sabe que tem que mudar, mas reluta em fazê-lo.

Migrar de uma programação ladder ou FSC em CLP para uma programação de sistemas multi-agentes também é um paradigma para os desenvolvedores de sistemas de manufatura na indústria. O desenvolvimento de uma ferramenta que se proponha a executar a migração destas linguagens é algo que auxiliaria muito e se propõe como continuidade do trabalho.

Neste trabalho foram abordados dispositivos e equipamentos cuja aplicação foi a realização da manufatura ou transporte, porém o conceito de sistemas produtivos orientados a serviços e sistemas multi-agentes abre espaço para desenvolvimento de sistemas que possam realizar inspeção, promover controle de qualidade, promover o momento de realizar manutenções preditivas, entre outros serviços. E também se pode pensar em um agente que modifica física e computacionalmente outro agente, em função de uma demanda ou situação de produção. É a capacidade do sistema se transformar (evoluir) para atender outros requisitos, isto numa esfera de *hardware* do sistema.

Como trabalhos futuros se propõe a implantação física do protótipo demonstrador, que teria cunho didático, auxiliando alunos no seu aprendizado e fornecendo dados para análise de trabalhos e conceitos nestes sistemas.

Também se propõe à concepção de sistema multi-agente com arquitetura orientada a serviços que possa inspecionar outros agentes e auxiliá-lo na sua transformação (evolução) para compor outro serviço que originalmente não possuía.

# REFERÊNCIAS

ALSTERMAN, H.; ONORI, M. Definitions, limitations and approaches of evolvable assembly system platforms. **IFIP International Federation For Information Processing**, Boston, v. 159, p. 367–377. 2004.

BARATA, J. et al. Integrated And Distributed Manufacturing: A Multi-Agent Perspective. In: WORKSHOP ON EUROPEAN SCIENTIFIC AND INDUSTRIAL COLLABORATION - WESIC, 3., 2001, Enschede. **Proceedings…** Enschede : WESIC, 2001. p. 27-29. Disponível em: <http://www.ipb.pt/~pleitao/papers/wesic2001.pdf >. Acesso em: 12 abr. 2011.

BARATA, J.; MENDES, P.; RIBEIRO, L. MAS And SOA: Complementary Automation Paradigms. In: IFIP INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY FOR BALANCED AUTOMATION SYSTEMS, 8., 2008, Porto. **Proceedings…** Porto: IFIP, 2008 . Disponível em: <http://www.springerlink.com/content/g75735487077g73x/fulltext.pdf>. Acesso em: 28 mar. 2011.

BARATA, J.; RIBEIRO, L.; COLOMBO, A. Diagnosis using Service Oriented Architectures (SOA). In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS - INDIN, 5., 2007, Viena. **Proceedings . . .** Viena: IEEE, 2007. v.2, p.1203-1208. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4384902&isnumber=4384837>. Acesso em: 28 mar. 2011.

BARROS, M. R. A. **Estudo da automação de células de manufatura para montagens e soldagem industrial de carrocerias automotivas**. 2006. 146 f. Dissertação (Mestrado em Engenharia) - Escola Politécnica, Universidade de São Paulo - USP, São Paulo, 2006. Disponível em: < http://www.automotiva-poliusp.org.br/mest/banc/pdf/barros_marcelo.pdf>. Acesso em 4 jan. 2012.

BELLIFEMINE F.; CAIRE G.; GREENWOOD D. **Developing multi-agent systems with JADE** . West Sussex: John Wiley, 2004. 303 p. ISBN: 978-04-7005-747-6.

BUSSMANN, S. An Agent-Oriented Architecture for Holonic Manufacturing Control. In: WORKSHOP ON INTELLIGENT MANUFACTURING SYSTEMS - IFAC,1., 1998, Lausanne**. Proceedings . . .** Lausanne: IFAC, 1998. p. 1-12. Disponível em: < http://stefan-bussmann.de/downloads/ims98.pdf>. Acesso em: 12 dez. 2011.

CAVALCANTE, A.L.D. **Arquitetura baseada em agentes e auto-organiz ável para a manufatura**. 2012. 140 f. Tese (Doutorado em Engenharia Elétrica) – Programa Pós-graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Sul – UFRGS, Porto Alegre, 2012.

CAVALCANTE, A.; PEREIRA, C.E.; BARATA, J. Component-Based Approach to the Development of Self-X Automation Systems. In: IFAC WORKSHOP ON INTELLIGENT MANUFACTURING SYSTEMS, 10., 2010, Lisboa. **Proceedings...** Lisboa: IFAC, 2010. p. 239-244. Disponível em < http://www.ifac-papersonline.net/Detailed/43369.html>. Acesso em : 16 mar. 2011.

COLOMBO, A. W. et al. Service-Oriented Control Architecture for Reconfigurable Production Systems. In: THE IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS, 6., 2008, Daejeon. **Proceedings…** Daejeon: IEEE, 2008. p.13-16. Disponível em < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4618201&isnumber=4618046>. Acesso em: 15 mar. 2011.

DEITEL, P. J.; DEITEL, H. M. **Java how to program**. 7. ed. Upper Saddle River : Pearson Prentice Hall, 2007. 1596 p. ISBN: 978-01-3222-220-4

FERRARINI, L. et al. Control Functions Development For Distributed Automation Systems Using The Torero Approach. In: IFAC WORLD CONGRESS, 16., 2005, Praga. **Proceedings…** Praga: IFAC, 2005. Disponível em: <http://www.nt.ntnu.no/users/skoge/prost/proceedings/ifac2005/Fullpapers/04250.pdf>. Acesso em: 16 abr. 2011.

FERRARINI, L.; VEBER, C.; LORENTZ, K. A case study for modeling and design of distributed automation systems. In: IEEE - INTERNATIONAL CONFERENCE ON ADVANCED INTELIGENT MECHATRONICS, 1. , 2003, Kobe. **Proceedings . . .** Kobe: IEEE, 2003. p. 1043-1048. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1225486&isnumber=27509>. Acesso em: 28 mar. 2011.

FERREIRA, B. et al. An architecture for self-managing evolvable assembly systems. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETIC, 2009, San Antonio. **Proceedings…** San Antonio: IEEE, 2009. p. 2707-2712. Disponível em: < http://www.reginafrei.ch/pdf/SMC2009-frei-ferreira-gdm-jab-v3.pdf>. Acesso em 3 set. 2011.

FESTO Didatic WEB side. MicroFMS and MultiFMS. 2011. Disponível em: < http://www.festo-didactic.com/int-en/learning-systems/learning-factories,cim-fms-systems/microfms-and-multifms.htm?fbid=aW50LmVuLjU1Ny4xNy4xOC42MzguNzY0MQ>. Acesso em 16 ago. 2011.

FIPA Fundation. The Fundation for Inteligent Physical Agents. Disponível em: <http://www.fipa.org>. Acesso em: 20 mai. 2011.

FLEISCHHAUER, L. I. A. **O uso da tecnologia de agentes na integração da Programação da Produção**. 1996. 102f. Dissertação (Mestrado em Engenharia) - Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina - UFSC. Florianópolis, 1996. Disponível em: < http://eps.ufsc.br/disserta97/amaral/>. Acesso em: 18 jan. 2012.

FLORES, A.P. et al. Quantitative Evaluation of Distributed Object-Oriented Programming Environments for Real-Time Applications. In: IEEE INTERNATIONAL SYMPOSIUM ON OBJECT-ORIENTED REAL-TIME DISTRIBUTED COMPUTING, 2., 1999, Saint-Malo. **Proceedings…** Saint-Malo: IEEE, 1999. p. 133-138. Disponível em:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=776366>. Acesso em: 10 jan. 2012.

FRANCHI, C. M.; CAMARGO, V. L. A. **Controladores lógicos programáveis**: sistema discretos. 2.ed. São Paulo: Érica, 2011. 352 p.

GEORGINI, M. **Automação aplicada**: descrição e implementação de sistemas sequenciais com PLCs. São Paulo: Érica, 2000. 216 p. ISBN 85-7194-724-4

GU, W. et al. A neuroendocrine-inspired bionic manufacturing system. **Journal Of Systems Science And Systems Engineering**, Pequim, v. 20, n. 3, p. 275– 293. 2011.

GUIMARÃES, H. C. F. **Norma IEC61131-3 para programação de controladores programáveis:** Estudo e aplicação. 2005. 86f. Monografia (Trabalho de conclusão do Curso de Engenharia Elétrica) – Centro Tecnológico, Universidade Federal do Espírito Santo, Vitória , 2005. Disponível em: <http://www2.ele.ufes.br/~projgrad/documentos/PG2005_1/hugocasatiferreiraguimaraes.pdf> . Acesso em 22 mar. 2012.

HERRERA, V.V. et al. Integration of Multi-Agent Systems and Service-Oriented Architecture for Industrial Automation. In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS – INDIN, 6., 2008, Daejeon. **Proceedings…** Daejon: IEEE, 2008. p.768-773. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4618205&isnumber=4618046>. Acesso em: 14 abr. 2011.

JAMES F.; SMIT, H. Service-oriented paradigms in industrial automation. **IEEE transactions on industrial informatics**, Nova York, v. 1, n. 1, p. 62-70, 2005. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1411764>. Acesso em: 13 nov. 2011. ISSN: 1551-3203.

JÓZEFOWSKA, J. **Just-in-Time Scheduling:** Models and Algorithms for Computer and Manufacturing Systems. Norwell: LLC, 2007. 255 p. ISBN: 978-03-8771-718-0.

KUSIAK, A. **Modelling and design of flexible manufacturing systems**. Amsterdam: North-holland, 1986. 431 p. ISBN: 04-4442-596-9.

LEITÃO, P. et al. Trends In Agile And Co-Operative Manufacturing. In: LOW COST AUTOMATION SYMPOSIUM - LCA, 2001, Berlin. **Proceedings…** Berlin: IFAC, 2001. p. 156-165. Disponível em: <http://hdl.handle.net/10198/1426>. Acesso em: 12 abr. 2011.

LEITÃO, P.; RESTIVO, F. An agile and cooperative architecture for distributed manufacturing systems. In: INTERNATIONAL CONFERENCE ROBOTICS AND MANUFACTURING - IASTED, 2001, Cancun. **Proceedings…** Cancun: IASTED, 2001. p. 21-24. Disponível em: <http://bibliotecadigital.ipb.pt/bitstream/10198/1467/1/2001-RM.pdf>. Acesso em: 12 nov. 2011.

LORENTZ, K. et al. Next Generation Integrated Development of Automation Control Code in TORERO. In: IEEE INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS – ISIE, 2003, Rio de Janeiro. **Proceedings…** Rio de Janeiro: IEEE, 2003. v.

2, p. 858-861. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1267933&isnumber=28352>. Acesso em: 16 abr. 2011.

MENDES, J.M. et al. Service-oriented Process Control using High-Level Petri Nets. In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS – INDIN, 6., 2008, Daejeon. **Proceedings…** Daejeon: IEEE, 2008. p. 750-755. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4618202&isnumber=4618046>. Acesso em: 15 mar. 2011.

MONTGOMERY, D.C. **Design and Analysis of Experiments**. 5. ed. New York : John Wiley, 2001. 684 p. ISBN 04-7131-649-0.

NILSSON, K. ; PIRES, J.N.; VEIGA, G. On The Use Of Service Oriented Software Platforms For Industrial Robotic Cells. In: IFAC INTERNATIONAL WORKSHOP INTELLIGENT MANUFACTURING SYSTEMS, 8., 2007, Alicante. **Proceedings…** Alicante: Universidade de Alicante, 2007. Disponível em: <http://www.smerobot.org/08_scientific_papers/papers/Pires_IMS2007_UPnP4_gv_jnp_kn_v5_ADDF.pdf>. Acesso em: 12 abr. 2011.

OHNO, T. **Sistema toyota de produção: alem da produçao em larga escala.** Porto Alegre: Bookman, 1997. 152 p. ISBN: 978-85-7307-170-2.

OLIVEIRA, J. A. B. **Coalition based approach for shop floor agility**. 2003. 329 f. Tese (PhD em Engenharia Elétrica) - Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa - UNINOVA, Lisboa, 2003. Disponível em: <http://run.unl.pt/bitstream/10362/2483/1/Oliveira_2004.pdf>. Acesso em 6 jan. 2012.

ONORI, M. et al. Evolvable Assembly Systems Basic Principles. **Information Technology for Balanced Manufacturing Systems**, Boston, v. 220, p. 317–328, 2006.

OROZCO, O. J.; LASTRA, J. L. M. Adding Function Blocks of IEC 61499 Semantic Description to Automation Objects. In: IEEE CONFERENCE OF EMERGING TECHNOLOGIES AND FACTORY AUTOMATION, 11., 2006, Praga. **Proceedings…** Praga: IEEE, 2006, p. 537–544.
Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4178288>. Acesso em: 20 fev. 2011.

PEREIRA, C. E.; CARRO, L. Distributed real-time embedded systems: Recent advances, future trends and their impact on manufacturing plant control. **Annual Reviews In Control**, Oxford, v. 31, p. 81–92, 2007.

QUINN, R. D. et al. An agile manufacturing workcell design. **IIE Transactions**. [S. l.], v. 29, n.10, 1997, p. 901-909. 1997. Disponível em <http://www.tandfonline.com/doi/abs/10.1080/07408179708966410>. Acesso em 12 de abr. 2012.

REN, L.; TAO, F.; ZHANG, L. A Virtual SOA Model for Network Manufacturing Integration. In: THE INTERNATIONAL CONFERENCE ON ELECTRICAL AND CONTROL ENGENEERING - ICECE, 2010, Wuhan. **Proceedings . . .** Wuhan: ICECE,

2010. p. 3570-3573. Disponível em: <http://www.docstoc.com/docs/64300177/Research-on-Soft-Start-Up-of-Vsc-Hvdc>. Acesso em: 18 abr. 2011.

REN, Y.; LUQING, Y.; CHUANG, F. A Multi-Agent-Based Remote Maintenance Support and Management System. In: IEEE/WIC/ACM INTERNATIONAL ON INTELLIGENT AGENT TECHNOLOGY – IAT, 2004, Pequim. **Proceedings…** Pequim : IEEE, 2004. p. 496-499. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1343004&isnumber=29567>. Acesso em: 12 jan. 2012.

RESTIVO, F. J.; LEITÃO, P. Implementation of a holonic control system in a flexible manufacturing system. **IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS**, [S. l.], v. 38, n. 5, p. 699-709, 2008. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4603100&tag=1>. Acesso em: 11 jan. 2012.

SATO, D. T. **Uso eficaz de métricas em métodos ágeis de desenvolvimento de software**. 2007, 155 f. Dissertação (Mestrado em ciência da computação) – Instituto e Matemática e Estatística, Universidade de São Paulo, São Paulo, 2007. Disponível em: < http://www.teses.usp.br/teses/disponiveis/45/45134/tde-06092007-225914/pt-br.php>. Acesso em: 13 fev. 2012.

SCHWAB, C. et al. Mapping of IEC 61499 Function Blocks to Automation Protocols within the TORERO Approach. In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS, 2., 2004, Berlin. **Proceedings…** Berlin: IEEE, 2004. p. 149-154. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1417319>. Acesso em: 16 abr. 2011.

SHINGO, S., **O sistema toyota de produção : do ponto de vista da engenharia de produção**. Porto Alegre: Bookman, 1996. 291 p.

SILVA, L. A. M. **Estudo e Desenvolvimento de Sistemas Multiagentes usando JADE: Java Agent Development framework**. 2003. 97 f. Monografia ( Trabalho de conclusão de curso de Bacharel em Informática) - Centro de Ciências Tecnológicas, Universidade de Fortaleza – UNIFOR , Fortaleza , 2003. Disponível em: < http://jade.cselt.it/papers/2003/monografia.pdf>. Acesso em 22 dez. 2011.

SILVEIRA, P. R.; SANTOS, W. E. **Automação e controle discreto.** 2. ed. São Paulo: Érica, 1999. 229 p.

SODA Consorsium. Service Oriented Device & Delivery Architecture. Disponível em: <http://www.soda-itea.org/>. Acesso em: 13 abr. 2011.

THARUMARAJAH, A.; WELLS, A.J.; NEMES, L. Comparison of emerging manufacturing concepts. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS, 1998, San Diego. **Proceedings…** San Diego: IEEE , 1998. p. 325–331. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=725430&isnumber=15656>. Acesso em: 24 nov. 2011.

TILAB Group. JAVA Agent Development Framework. Disponível em: <http://jade.tilab.com>. Acesso em: 25 mai. 2011.

TORERO Consorsium WEB side. TORERO Total life cycle web-integrated control. Disponível em: <http://www.uni-magdeburg.de/iaf/cvs/torero/>. Acesso em 16 mai. 2011.

UEDA, K. A concept for bionic manufacturing systems based on DNA-type information. In: INTERNATIONAL PROLAMAT: CONFERENCE ON HUMAN ASPECTS IN COMPUTER INTEGRATED MANUFACTURING, 8., 1992a, Holanda. **Proceedings…** Holanda: North-Holland Publishing Co., 1992. p. 853–863.

UEDA, K. Emergent synthesis approaches to biological manufacturing systems. In: DIGITAL ENTERPRISE TECHNOLOGY - DET. 1992b, Setubal. **Proceedings…** Setubal: Springer US, 1992. p. 25–34.

VRBA, P.; MARIK, V. Capabilities of Dynamic Reconfiguration of Multiagent-Based Industrial Control Systems. **IEEE Transactions On Systems, Man, And Cybernetics—Part A: Systems And Humans**. [S. l.], v. 40, n. 2, 2010, p. 213-223. Disponível em < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5353631&isnumber=5415834>. Acesso em 15 dez. 2011.

XUEMEI, H. et al. Theoretical analyze and implementation method of reconfigurable assembly line based on agent and holon**. Fifth World Congress on Intelligent Control and Automation**, v. 3, 2004, p. 2830- 2833. Disponível em <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1342116&isnumber=29572>. Acesso em 15 jan. 2012.

WOMACK, James P.; JONES, Daniel T.; ROOS, Daniel. **A máquina que mudou o mundo**. Tradução Ivo Korytowski. 10. Ed. Rio de Janeiro: Elsevier, 2004. 333 p. ISBN: 85-352-1269-8.

ZYLSTRA, K. D. **Lean distribution: applying Lean manufacturing to distribution, logistics, and supply chain**. Hoboken: John Wiley, 2006. 223 p. ISBN: 978-04-7174-075-9.